

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 7,930,576 C1  
APPLICATION NO. : 90/011754  
DATED : September 4, 2012  
INVENTOR(S) : Ian F. Harris and Drew J. Dutton

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

**In the Claims:**

Column 12

Lines 24-25, in claim 181, delete "host memory and" and insert -- host memory bus and --, therefor.

Signed and Sealed this  
Fifteenth Day of October, 2013



Teresa Stanek Rea  
*Deputy Director of the United States Patent and Trademark Office*

***EX PARTE* REEXAMINATION**

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

|                             |  |                                 |                 |                 |
|-----------------------------|--|---------------------------------|-----------------|-----------------|
| Reexamination Control. No.: | 90/011,754   | §                               | Atty.Dkt.No.:   | 5707-26200      |
| Inventor:                   | Ian F. Harris, et al.  | §                               | Examiner:       | Peikari, Behzad |
| Request Filing Date:        | June 21, 2011  | §                               | Group/Art Unit: | 3992            |
| U.S. Patent No.:            | 7,930,576  | §                               | Conf. No.:      | 4071            |
| Title:                      | SHARING NON-SHARABLE<br>DEVICES BETWEEN AN<br>EMBEDDED CONTROLLER<br>AND A PROCESSOR IN A<br>COMPUTER SYSTEM | §<br>§<br>§<br>§<br>§<br>§<br>§ |                 |                 |

**TRANSMITTAL LETTER**

Transmitted herewith for filing in the captioned case is the following:

- (1) Form PTO-1050. Errors which occur at important points in the captioned patent or which may otherwise affect the understanding or interpretation of the patent are thereon corrected.

All of the errors shown in PTO-1050 are due to Patent Office oversights. A Certificate of Correction is requested under 35 U.S.C. § 254. Applicants believe that no fees are required, however, should any fees be required, please deduct them from Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 501505/5707-26200/JCH.

Respectfully submitted,

/Jeffrey C. Hood/  
Jeffrey C. Hood, Reg. #35198

Meyertons, Hood, Kivlin, Kowert & Goetzel PC  
P.O. Box 398  
Austin, Texas 78767-0398  
(512) 853-8800  
DATE: 2012-09-26 JCH/kmm

UNITED STATES PATENT AND TRADEMARK OFFICE

**CERTIFICATE OF CORRECTION**

PATENT NO.: 7,930,576 C1

DATED: September 4, 2012

INVENTOR(S): Ian F. Harris and Drew J. Dutton

It is certified that an error or errors appear in the above-identified patent and that said EX PARTE REEXAMINATION CERTIFICATE is hereby corrected as shown below:

**In the Claims:**

Column 12

Lines 24-25, in claim 181, delete “host memory and” and insert -- host memory bus and --, therefor.

-----  
MAILING ADDRESS OF SENDER:

Jeffrey C. Hood, Esq.,  
Meyertons, Hood, Kivlin, Kowert & Goetzel PC  
P.O. Box 398  
Austin, Texas 78767-0398

PATENT NO. 7,930,576 C1

**Certificate of Correction (PTO Form 1050)**

| Electronic Acknowledgement Receipt   |  |
|--------------------------------------|--|
| EFS ID:                              | 13839932   |
| Application Number:                  | 90011754   |
| International Application Number:    |  |
| Confirmation Number:                 | 4071   |
| Title of Invention:                  | SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM |
| First Named Inventor/Applicant Name: | 7,930,576 B2   |
| Customer Number:                     | 35690  |
| Filer:                               | Jeffrey C Hood/Karen Miller  |
| Filer Authorized By:                 | Jeffrey C Hood   |
| Attorney Docket Number:              | JCLA38862-RE   |
| Receipt Date:                        | 26-SEP-2012  |
| Filing Date:                         | 21-JUN-2011  |
| Time Stamp:                          | 11:44:18   |
| Application Type:                    | Reexam (Patent Owner)  |

### Payment information:

|                        |    |
|------------------------|----|
| Submitted with Payment | no |
|------------------------|----|

### File Listing:

| Document Number | Document Description                  | File Name  | File Size(Bytes)/<br>Message Digest               | Multi Part /.zip | Pages (if appl.) |
|-----------------|---------------------------------------|--|---|------------------|------------------|
| 1               | Request for Certificate of Correction | 5707-26200_transmittal_letter_correction_request_filed.pdf | 73453<br>da29e78802190a15050acd5e9a384746d1715bca | no               | 1                |

### Warnings:

### Information:



|  |                                       |                                      |   |        |   |
|--|---------------------------------------|--------------------------------------|---|--------|---|
| 2  | Request for Certificate of Correction | 5707-26200_cert_correction_filed.pdf | 61576<br>0f2f69fd149e906d76d58396df1b69d9053a7bfa | no     | 1 |
| <b>Warnings:</b>   |                                       |                                      |   |        |   |
| <b>Information:</b>  |                                       |                                      |   |        |   |
| <b>Total Files Size (in bytes):</b>  |                                       |                                      |   | 135029 |   |
| <p><b>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</b></p> <p><b><u>New Applications Under 35 U.S.C. 111</u></b><br/> If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</p> <p><b><u>National Stage of an International Application under 35 U.S.C. 371</u></b><br/> If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</p> <p><b><u>New International Application Filed with the USPTO as a Receiving Office</u></b><br/> If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.</p> |                                       |                                      |   |        |   |



US007930576C1

(12) **EX PARTE REEXAMINATION CERTIFICATE** (9280th)

**United States Patent**

**Harris et al.**

(10) **Number:** **US 7,930,576 C1**

(45) **Certificate Issued:** **Sep. 4, 2012**

(54) **SHARING NON-SHARABLE DEVICES  
BETWEEN AN EMBEDDED CONTROLLER  
AND A PROCESSOR IN A COMPUTER  
SYSTEM**

(75) Inventors: **Ian F. Harris**, Kings Park, NY (US);  
**Drew J. Dutton**, Austin, TX (US)

(73) Assignee: **Standard Microsystems Corporation**,  
Hauppauge, NY (US)

**Reexamination Request:**

No. 90/011,754, Jun. 21, 2011

**Reexamination Certificate for:**

Patent No.: **7,930,576**  
Issued: **Apr. 19, 2011**  
Appl. No.: **11/958,601**  
Filed: **Dec. 18, 2007**

**Related U.S. Application Data**

(60) Provisional application No. 60/910,863, filed on Apr. 10, 2007.

(51) **Int. Cl.**

**G06F 1/26** (2006.01)  
**G06F 15/177** (2006.01)

(52) **U.S. Cl.** ..... **713/324; 713/100; 713/2**

(58) **Field of Classification Search** ..... None  
See application file for complete search history.

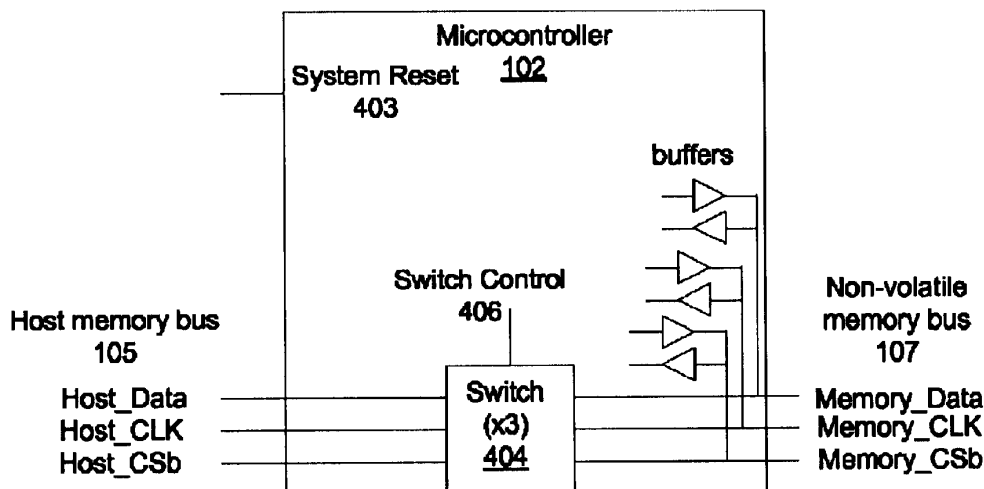
(56) **References Cited**

To view the complete listing of prior art documents cited during the proceeding for Reexamination Control Number 90/011,754, please refer to the USPTO's public Patent Application Information Retrieval (PAIR) system under the Display References tab.

*Primary Examiner*—B. James Peikari

(57) **ABSTRACT**

System and method for sharing a device, e.g., non-volatile memory, between a host processor and a microcontroller. In response to system state change to a first state wherein the microcontroller is assured safe access to the non-volatile memory (e.g., in response to power-on reset, system reset, sleep state, etc.), the microcontroller holds the system in the first state (e.g., system reset), and switches access to the non-volatile memory from the processor to the microcontroller. While the system is held in the first state, the microcontroller accesses the device (e.g., non-volatile memory), e.g., fetches program instructions/data from the non-volatile memory and loads the program instructions/data into a memory of the microcontroller. After the access, the microcontroller changes or allows change of the system state, e.g., switches access to the device, e.g., the non-volatile memory, from the microcontroller to the processor, and releases the system from the first state.



**1**  
**EX PARTE**  
**REEXAMINATION CERTIFICATE**  
**ISSUED UNDER 35 U.S.C. 307**

THE PATENT IS HEREBY AMENDED AS  
INDICATED BELOW.

**Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.**

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims 1-23 is confirmed.

New claims 24-213 are added and determined to be patentable.

24. *The system of claim 1, wherein the change in system state to the first state is performed by hardware.*

25. *The system of claim 1, wherein the change in system state to the first state is in response to a hardware signal.*

26. *The system of claim 1, wherein the change in system state to the first state is in response to a software event.*

27. *The system of claim 1, wherein the change in system state to the first state is in response to a token.*

28. *The system of claim 1, wherein the change in system state to the first state is in response to the system passing the microcontroller a token which grants the microcontroller ownership of the memory through a software handshake.*

29. *The system of claim 1, wherein the change in system state to the first state is in response to a software condition.*

30. *The system of claim 1, wherein the change in system state to the first state is in response to a software message.*

31. *The system of claim 1, wherein the change in system state to the first state is in response to a software handshake.*

32. *The system of claim 1, wherein the first state comprises a power on reset (POR) state.*

33. *The system of claim 1, wherein the first state comprises a system reset state.*

34. *The system of claim 1, wherein the first state comprises a sleep state.*

35. *The system of claim 1, wherein to hold the system in the first state, the microcontroller is configured to:*

*in response to a power on reset, hold a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory.*

36. *The system of claim 35, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller while the system is held in the first state, the microcontroller is configured to:*

*fetch the program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller while the system reset signal is held in the reset state.*

37. *The system of claim 35, wherein to switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state after the program instructions or data have been loaded, the microcontroller is configured to:*

*after the program instructions or data have been loaded, permit the processor in the system access to the non-volatile memory, and release the system reset signal from the reset state.*

**2**

38. *The system of claim 1, wherein the first state comprises a state wherein it is known by the microcontroller that the system cannot or will not access the device.*

39. *The system of claim 1, wherein the microcontroller is included within a system-on-chip (SoC).*

40. *The system of claim 1, wherein the microcontroller comprises a keyboard controller for managing or controlling communications between a keyboard and the processor.*

41. *The system of claim 40, wherein the non-volatile memory stores keyboard controller (KBC) code.*

42. *The system of claim 1, wherein the program instructions or data are usable by the microcontroller for startup, general, or reset functionality.*

43. *The system of claim 1, wherein the program instructions or data are usable by the microcontroller for initializing the microcontroller.*

44. *The system of claim 1, wherein during system shutdown, the microcontroller is further configured to:*

*configure the non-volatile memory to allow access to the non-volatile memory only by the microcontroller before power-off;*

*queue one or more writes or other exchanges with the non-volatile memory; and*

*complete the one or more writes or other exchanges with the non-volatile memory before final power-off.*

45. *The system of claim 1, further comprising:*

*a device interface coupled to the non-volatile memory, wherein the device interface controls access to the non-volatile memory, wherein during system shutdown, the microcontroller is further configured to:*

*redirect the device interface to allow access to the non-volatile memory only by the microcontroller before power-off;*

*queue one or more writes or other exchanges with the non-volatile memory; and*

*complete the one or more writes or other exchanges with the non-volatile memory before final power-off.*

46. *The system of claim 1, further comprising:*

*one or more additional processors, each coupled to the microcontroller and the non-volatile memory, and wherein the microcontroller has control over the power or execution states of the processor and the one or more additional processors for sharing the non-volatile memory between the plurality of processors.*

47. *The system of claim 1, further comprising:*

*one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, wherein each additional microcontroller is configured to share the non-volatile memory with the processor.*

48. *The system of claim 1, further comprising:*

*one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, wherein each additional microcontroller is configured to share the non-volatile memory, and wherein each microcontroller is configured to suspend or prevent access to the non-volatile memory by other microcontrollers and the processor while it accesses the non-volatile memory.*

49. *The system of claim 1, further comprising:*

*a system interface, coupled to the processor, wherein the microcontroller is situated between the system interface and the non-volatile memory, and wherein the microcontroller is configured to intercept communications between the system interface and the non-volatile memory; and*

*a device interface, included in or associated with the non-volatile memory, wherein the device interface is config-*

3

ured to direct or restrict access to the non-volatile memory by the processor or the microcontroller.

50. The system of claim 49, wherein the microcontroller is configured to set one or more control signals to configure or redirect the device interface to grant exclusive access to the non-volatile memory to the microcontroller.

51. The system of claim 1, wherein the non-volatile memory comprises flash memory.

52. The system of claim 1, wherein the change in system state to the first state is in response to a management signal, wherein to hold the system in the first state, the microcontroller is further configured to hold the management signal in a corresponding state, or to hold or assert another management signal in a specified state.

53. The system of claim 52, further comprising:

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured by the microcontroller to direct or restrict access to the non-volatile memory by the processor or the microcontroller independent of the management signal.

54. The system of claim 52, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller via a system management bus configured for communicating management signals between the system interface and the microcontroller.

55. The system of claim 52, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller via a system interface bus configured for communicating management signals between the system interface and the microcontroller.

56. The system of claim 52, wherein the management signal or the another management signal is invoked by the microcontroller.

57. The system of claim 52, wherein the management signal or the another management signal is not invoked by the microcontroller.

58. The system of claim 52, wherein the management signal or the another management signal is a software signal.

59. The system of claim 52, wherein the management signal or the other management signal is a hardware signal.

60. The system of claim 1, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via software.

61. The system of claim 1, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via hardware.

62. The system of claim 1, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via a combination of software and hardware.

63. The system of claim 1, wherein the change in system state to the first state is not dependent on a management signal.

64. The system of claim 1, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller, the microcontroller is configured to:

fetch initialization code or data for the system.

65. The system of claim 64, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller, the microcontroller is further configured to:

fetch program instructions or data related to pre-boot security operations; and

4

perform the pre-boot security operations prior to boot-up of the system.

66. The system of claim 1, wherein the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus.

67. The system of claim 66, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable to switchably intercept or pass communications between the processor and the non-volatile memory.

68. The system of claim 67, wherein the host memory bus and the non-volatile memory bus are SPI (serial peripheral interface) buses.

69. The system of claim 67, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor and the non-volatile memory.

70. The system of claim 69, wherein the switch comprises a Q switch or a conceptual Q switch.

71. The system of claim 69, wherein to switchably intercept or pass communications between the processor and the non-volatile memory, the microcontroller is configured to place the switch into a high-impedance state.

72. The system of claim 69, wherein to switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state, the microcontroller is configured to:

reconfigure the access to the non-volatile memory via the switch, and release the system from the first state, thereby permitting the processor access to the non-volatile memory.

73. The system of claim 69,

wherein the first state comprises a reset state, wherein to hold the system in the first state, the microcontroller is configured to maintain assertion of a system reset signal; and

wherein to switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state, the microcontroller is configured to:

terminate a prior suspension or preemption of communications between the host interface and the non-volatile memory via the switch, thereby allowing resumption of the communications; and de-assert the system reset signal.

74. The system of claim 67, wherein the host memory bus and the non-volatile memory bus each comprises:

a data line;

a clock line; and

a CSb (Chip Select bar) line.

75. The system of claim 67, wherein the host memory bus and the non-volatile memory bus each comprises a plurality of lines.

76. The system of claim 75, wherein each line of the host memory bus and the non-volatile memory bus is independently switchable.

77. The system of claim 75, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller for communicating signals between the system interface and the microcontroller;

wherein the lines of the host memory bus and the non-volatile memory bus are switchable together to suspend

5

or inhibit communications between the system interface and the non-volatile memory.

78. The system of claim 75, wherein the lines of the host memory bus and the non-volatile memory bus are coupled to respective buffers for communicating data or signals between the non-volatile memory bus and the microcontroller.

79. The system of claim 78, further comprising:

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured by the microcontroller to:

when the system is held in the first state, preclude the processor from accessing the non-volatile memory.

80. The system of claim 79, wherein the microcontroller is further configured to:

once the microcontroller has completed fetching program instructions or data from the non-volatile memory, or has determined that it is no longer assured of safe access to the non-volatile memory, reconfigure the device interface via control signals to enable access by the processor to the non-volatile memory.

81. The system of claim 1,

wherein the first state comprises a reset state, wherein to hold the system in the first state, the microcontroller is configured to maintain assertion of a system reset signal; and

wherein to switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state, the microcontroller is configured to:

terminate a prior suspension or preemption of communications between the processor and the non-volatile memory, thereby allowing communications to occur between the processor and the non-volatile memory; and

de-assert the system reset signal.

82. The system of claim 1, further comprising:

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured by the microcontroller to:

when the system is held in the first state, preclude the processor from accessing the non-volatile memory.

83. The system of claim 1, wherein the microcontroller is further configured to:

while the system is held in the first state, exchange data or program instructions with the non-volatile memory.

84. The system of claim 83, wherein to exchange data or program instructions with the non-volatile memory, the microcontroller is configured to:

invoke a read or write by another component of the system.

85. The system of claim 1, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller via a system management bus configured for communicating management signals between the system interface and the microcontroller;

wherein the microcontroller comprises a system reset line for communicating a reset signal or other signal to or from the system interface.

86. The system of claim 85, wherein the reset signal or other signal corresponds with or invokes a suspension of non-volatile memory access by the processor.

87. The system of claim 85, wherein the reset signal or other signal indicates a system state in which the microcontroller has safe access to the non-volatile memory.

6

88. The system of claim 85, wherein to hold the system in the first state, the microcontroller is configured to hold the system reset signal or other signal in a reset state.

89. The method of claim 6, wherein the change in system state to the first state is performed by hardware.

90. The method of claim 6, wherein the change in system state to the first state is in response to a hardware signal.

91. The method of claim 6, wherein the change in system state to the first state is in response to a software event.

92. The method of claim 6, wherein the change in system state to the first state is in response to a token.

93. The method of claim 6, wherein the change in system state to the first state is in response to the system passing the microcontroller a token which grants the microcontroller ownership of the memory through a software handshake.

94. The method of claim 6, wherein the change in system state to the first state is in response to a software condition.

95. The method of claim 6, wherein the change in system state to the first state is in response to a software message.

96. The method of claim 6, wherein the change in system state to the first state is in response to a software handshake.

97. The method of claim 6, wherein the first state comprises a power on reset (POR) state.

98. The method of claim 6, wherein the first state comprises a system reset state.

99. The method of claim 6, wherein the first state comprises a sleep state.

100. The method of claim 6, wherein holding the system in the first state comprises, in response to a power on reset, holding a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory.

101. The method of claim 100, wherein said fetching program instructions or data from the non-volatile memory and loading the program instructions or data into a memory of the microcontroller while the system is held in the first state occur while the system reset signal is held in the reset state.

102. The method of claim 100, wherein said switching access to the non-volatile memory from the microcontroller to the processor comprises permitting the processor in the system to access the non-volatile memory, and wherein releasing the system from the first state after the program instructions or data have been loaded comprises releasing the system reset signal from the reset state.

103. The method of claim 6, wherein the first state comprises a state wherein it is known by the microcontroller that the system cannot or will not access the device.

104. The method of claim 6, wherein the microcontroller is included within a system-on-chip (SoC).

105. The method of claim 6, wherein the microcontroller comprises a keyboard controller for managing or controlling communications between a keyboard and the processor.

106. The method of claim 105, wherein the non-volatile memory stores keyboard controller (KBC) code.

107. The method of claim 6, wherein the program instructions or data are usable by the microcontroller for startup, general, or reset functionality.

108. The method of claim 6, wherein the program instructions or data are usable by the microcontroller for initializing the microcontroller.

109. The method of claim 6, further comprising:

during system shutdown:

configuring the non-volatile memory to allow access to the non-volatile memory only by the microcontroller before power-off;

queuing one or more writes or other exchanges with the non-volatile memory; and

completing the one or more writes or other exchanges with the non-volatile memory before final power-off.

7

110. The method of claim 6, further comprising:  
during system shutdown:

redirecting a device interface to allow access to the  
non-volatile memory only by the microcontroller  
before power-off, wherein the device interface is  
coupled to the non-volatile memory and controls  
access to the non-volatile memory;  
queuing one or more writes or other exchanges with the  
non-volatile memory; and  
completing the one or more writes or other exchanges  
with the non-volatile memory before final power-off.

111. The method of claim 6, wherein the system further  
includes one or more additional processors, each coupled to  
the microcontroller and the non-volatile memory, and  
wherein the method further includes controlling the power  
or execution states of the processor and the one or more  
additional processors for sharing the non-volatile memory  
between the plurality of processors.

112. The method of claim 6, wherein the system further  
includes one or more additional microcontrollers, each  
coupled to the processor and the non-volatile memory, and  
wherein the method further includes each additional micro-  
controller sharing the non-volatile memory with the proces-  
sor.

113. The method of claim 6, wherein the system further  
includes one or more additional microcontrollers, each  
coupled to the processor and the non-volatile memory, and,  
wherein the method further includes:

each additional microcontroller sharing the non-volatile  
memory; and

each microcontroller suspending or preventing access to  
the non-volatile memory by other microcontrollers and  
the processor while it accesses the non-volatile  
memory.

114. The method of claim 6, further comprising:

intercepting communications between a system interface  
and the non-volatile memory, wherein the system inter-  
face is coupled to the processor, and wherein the micro-  
controller is situated between the system interface and  
the non-volatile memory; and

a device interface directing or restricting access to the  
non-volatile memory by the processor or the  
microcontroller, wherein the device interface is  
included in or associated with the non-volatile memory.

115. The method of claim 114, wherein the microcontrol-  
ler is configured to set one or more control signals to config-  
ure or redirect the device interface to grant exclusive access  
to the non-volatile memory to the microcontroller.

116. The method of claim 6, wherein the non-volatile  
memory comprises flash memory.

117. The method of claim 6, wherein the change in system  
state to the first state is in response to a management signal,  
wherein to hold the system in the first state, the microcontrol-  
ler is further configured to hold the management signal in a  
corresponding state, or to hold or assert another manage-  
ment signal in a specified state.

118. The method of claim 117, further comprising:

a device interface directing or restricting access to the  
non-volatile memory by the processor or the microcon-  
troller independent of the management signal, wherein  
the device interface is included in or associated with  
the non-volatile memory.

119. The method of claim 117, further comprising:

a system interface communicating management signals  
between the system interface and the microcontroller,  
wherein the system interface is coupled to the

8

processor, and further coupled to the microcontroller  
via a system management bus.

120. The method of claim 117, further comprising:

a system interface communicating management signals  
between the system interface and the microcontroller,  
wherein the system interface is coupled to the  
processor, and further coupled to the microcontroller  
via a system interface bus.

121. The method of claim 117, wherein the management  
signal or the another management signal is invoked by the  
microcontroller.

122. The method of claim 117, wherein the management  
signal or the another management signal is not invoked by  
the microcontroller.

123. The method of claim 117, wherein the management  
signal or the another management signal is a software sig-  
nal.

124. The method of claim 117, wherein the management  
signal or the other management signal is a hardware signal.

125. The method of claim 6, wherein switching access to  
the non-volatile memory from the processor to the microcon-  
troller and from the microcontroller to the processor is per-  
formed via software.

126. The method of claim 6, wherein switching access to  
the non-volatile memory from the processor to the microcon-  
troller and from the microcontroller to the processor is per-  
formed via hardware.

127. The method of claim 6, wherein switching access to  
the non-volatile memory from the processor to the microcon-  
troller and from the microcontroller to the processor is per-  
formed via a combination of software and hardware.

128. The method of claim 6, wherein the change in system  
state to the first state is not dependent on a management  
signal.

129. The method of claim 6, wherein said fetching pro-  
gram instructions or data from the non-volatile memory fur-  
ther comprises fetching initialization code or data for the  
system.

130. The method of claim 129, further comprising:

fetching program instructions or data related to pre-boot  
security operations; and

performing the pre-boot security operations prior to boot-  
up of the system.

131. The method of claim 6, wherein fetching the program  
instructions or data from the non-volatile memory occurs via  
a non-volatile memory bus that couples the microcontroller  
and the non-volatile memory.

132. The method of claim 131, wherein the microcontrol-  
ler is coupled to the processor via a host memory bus, and  
wherein the method further comprises the microcontroller  
switchably intercepting or passing communications between  
the processor and the non-volatile memory.

133. The method of claim 132, wherein the host memory  
bus and the non-volatile memory bus are SPI (serial periph-  
eral interface) buses.

134. The method of claim 132, wherein the microcontrol-  
ler comprises or is coupled to a switch, interposed between  
the non-volatile memory bus and the host memory bus, and  
wherein the microcontroller switchably intercepting or pass-  
ing communications comprises the microcontroller control-  
ling the switch to switchably intercept or pass communica-  
tions between the processor and the non-volatile memory.

135. The method of claim 134, wherein the switch com-  
prises a Q switch or a conceptual Q switch.

136. The method of claim 134, wherein switchably inter-  
cepting communications between the processor and the non-  
volatile memory comprises placing the switch into a high-  
impedance state.

9

137. The method of claim 134, wherein switching access to the non-volatile memory from the microcontroller to the processor comprises reconfiguring the access to the non-volatile memory via the switch.

138. The method of claim 134,

wherein the first state comprises a reset state, wherein holding the system in the first state comprises maintaining assertion of a system reset signal; and

wherein switching access to the non-volatile memory from the microcontroller to the processor comprises terminating a prior suspension or preemption of communications between the host interface and the non-volatile memory via the switch, thereby allowing resumption of the communications; and

wherein releasing the system from the first state comprises de-asserting the system reset signal.

139. The method of claim 132, wherein the host memory bus and the non-volatile memory bus each comprises:

a data line;

a clock line; and

a CSb (Chip Select bar) line.

140. The method of claim 132, wherein the host memory bus and the non-volatile memory bus each comprises a plurality of lines.

141. The method of claim 140, wherein each line of the host memory bus and the non-volatile memory bus is independently switchable.

142. The method of claim 140, further comprising:

switching the lines of the host memory bus and the non-volatile memory bus together to suspend or inhibit communications between a system interface and the non-volatile memory, wherein the system interface is coupled to the processor, and further coupled to the microcontroller for communicating signals between the system interface and the microcontroller.

143. The method of claim 140, wherein the lines of the host memory bus and the non-volatile memory bus are coupled to respective buffers for communicating data or signals between the non-volatile memory bus and the microcontroller.

144. The method of claim 143, further comprising:

when the system is held in the first state, precluding, by a device interface, the processor from accessing the non-volatile memory, wherein the device interface is included in or associated with the non-volatile memory.

145. The method of claim 140, further comprising:

once the microcontroller has completed fetching program instructions or data from the non-volatile memory, or has determined that it is no longer assured of safe access to the non-volatile memory, reconfiguring the device interface via control signals to enable access by the processor to the non-volatile memory.

146. The method of claim 6,

wherein the first state comprises a reset state, wherein holding the system in the first state comprises maintaining assertion of a system reset signal; and

wherein switching access to the non-volatile memory from the microcontroller to the processor comprises terminating a prior suspension or preemption of communications between the host interface and the non-volatile memory via the switch, thereby allowing resumption of the communications; and

wherein releasing the system from the first state comprises de-asserting the system reset signal.

10

147. The method of claim 6, further comprising:

when the system is held in the first state, precluding, by a device interface, the processor from accessing the non-volatile memory, wherein the device interface is included or associated with the non-volatile memory.

148. The method of claim 6, further comprising:

while the system is held in the first state, the microcontroller exchanging data or program instructions with the non-volatile memory.

149. The method of claim 148, wherein the microcontroller exchanging data or program instructions with the non-volatile memory comprises invoking a read or write by another component of the system.

150. The method of claim 6, further comprising:

communicating management signals between a system interface and the microcontroller, wherein the system interface is coupled to the processor and is further coupled to the microcontroller via a system management bus, and wherein the microcontroller comprises a system reset line for communicating a reset signal or other signal to or from the system interface.

151. The method of claim 150, wherein the reset signal or other signal corresponds with or invokes a suspension of non-volatile memory access by the processor.

152. The method of claim 150, wherein the reset signal or other signal indicates a system state in which the microcontroller has safe access to the non-volatile memory.

153. The system of claim 150, wherein holding the system in the first state comprises the microcontroller holding the system reset signal or other signal in a reset state.

154. The method of claim 6, wherein said holding the system in the first state is performed by the microcontroller.

155. The method of claim 6, wherein said switching access to the non-volatile memory from the processor to the microcontroller is performed by the microcontroller.

156. The method of claim 6, wherein said fetching program instructions or data from the non-volatile memory is performed by the microcontroller.

157. The method of claim 6, wherein said loading the program instructions or data into a memory of the microcontroller is performed by the microcontroller.

158. The method of claim 6, wherein said switching access to the non-volatile memory from the microcontroller to the processor is performed by the microcontroller.

159. The method of claim 6, wherein said releasing the system from the first state is performed by the microcontroller.

160. The system of claim 11, wherein the microcontroller is included within a system-on-chip (SoC).

161. The system of claim 11, wherein the microcontroller comprises a keyboard controller for managing or controlling communications between a keyboard and the processor.

162. The system of claim 161, wherein the non-volatile memory stores keyboard controller (KBC) code.

163. The system of claim 11, wherein the program instructions or data are usable by the microcontroller for startup, general, or reset functionality.

164. The system of claim 11, wherein the program instructions or data are usable by the microcontroller for initializing the microcontroller.

165. The system of claim 11, wherein during system shutdown, the microcontroller is further configured to:

configure the non-volatile memory to allow access to the non-volatile memory only by the microcontroller before power-off;

queue one or more writes or other exchanges with the non-volatile memory; and

11

complete the one or more writes or other exchanges with the non-volatile memory before final power-off.

166. The system of claim 11, further comprising:

a device interface coupled to the non-volatile memory, wherein the device interface controls access to the non-volatile memory, wherein during system shutdown, the microcontroller is further configured to:

redirect the device interface to allow access to the non-volatile memory only by the microcontroller before power-off;

queue one or more writes or other exchanges with the non-volatile memory; and

complete the one or more writes or other exchanges with the non-volatile memory before final power-off.

167. The system of claim 11, further comprising:

one or more additional processors, each coupled to the microcontroller and the non-volatile memory, and wherein the microcontroller has control over the power or execution states of the processor and the one or more additional processors for sharing the non-volatile memory between the plurality of processors.

168. The system of claim 11, further comprising:

one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, wherein each additional microcontroller is configured to share the non-volatile memory with the processor.

169. The system of claim 11, further comprising:

one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, wherein each additional microcontroller is configured to share the non-volatile memory, and wherein each microcontroller is configured to suspend or prevent access to the non-volatile memory by other microcontrollers and the processor while it accesses the non-volatile memory.

170. The system of claim 11, further comprising:

a system interface, coupled to the processor, wherein the microcontroller is situated between the system interface and the non-volatile memory, and wherein the microcontroller is configured to intercept communications between the system interface and the non-volatile memory; and

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured to direct or restrict access to the non-volatile memory by the processor or the microcontroller.

171. The system of claim 170, wherein the microcontroller is configured to set one or more control signals to configure or redirect the device interface to grant exclusive access to the non-volatile memory to the microcontroller.

172. The system of claim 11, wherein the non-volatile memory comprises flash memory.

173. The system of claim 11, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller, the microcontroller is configured to:

fetch initialization code or data for the system.

174. The system of claim 173, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller, the microcontroller is further configured to:

fetch program instructions or data related to pre-boot security operations; and

perform the pre-boot security operations prior to boot-up of the system.

175. The system of claim 11, wherein the microcontroller is coupled to the non-volatile memory via a non-volatile

12

memory bus, and wherein the microcontroller is operable to fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus.

176. The system of claim 175, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable to switchably intercept or pass communications between the processor and the non-volatile memory.

177. The system of claim 176, wherein the host memory bus and the non-volatile memory bus are SPI (serial peripheral interface) buses.

178. The system of claim 176, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor and the non-volatile memory.

179. The system of claim 178, wherein the switch comprises a Q switch or a conceptual Q switch.

180. The system of claim 178, wherein to switchably intercept or pass communications between the processor and the non-volatile memory, the microcontroller is configured to place the switch into a high-impedance state.

181. The system of claim 176, wherein the host memory and the non-volatile memory bus each comprises:

a data line;

a clock line; and

a CSb (Chip Select bar) line.

182. The system of claim 176, wherein the host memory bus and the non-volatile memory bus each comprises a plurality of lines.

183. The system of claim 182, wherein each line of the host memory bus and the non-volatile memory bus is independently switchable.

184. The system of claim 182, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller for communicating signals between the system interface and the microcontroller;

wherein the lines of the host memory bus and the non-volatile memory bus are switchable together to suspend or inhibit communications between the system interface and the non-volatile memory.

185. The system of claim 182, wherein the lines of the host memory bus and the non-volatile memory bus are coupled to respective buffers for communicating data or signals between the non-volatile memory bus and the microcontroller.

186. The system of claim 11, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller via a system management bus configured for communicating management signals between the system interface and the microcontroller;

wherein the microcontroller comprises a system reset line for communicating the system reset signal or other signal to or from the system interface.

187. The method of claim 17, wherein the microcontroller is included within a system-on-chip (SoC).

188. The method of claim 17, wherein the microcontroller comprises a keyboard controller for managing or controlling communications between a keyboard and the processor.

189. The method of claim 188, wherein the non-volatile memory stores keyboard controller (KBC) code.

190. The method of claim 17, wherein the program instructions or data are usable by the microcontroller for startup, general, or reset functionality.



13

191. The method of claim 17, wherein the program instructions or data are usable by the microcontroller for initializing the microcontroller.

192. The method of claim 17, further comprising:

during system shutdown;

configuring the non-volatile memory to allow access to the non-volatile memory only by the microcontroller before power-off;

queuing one or more writes or other exchanges with the non-volatile memory; and

completing the one or more writes or other exchanges with the non-volatile memory before final power-off.

193. The method of claim 17, further comprising:

during system shutdown;

redirecting a device interface to allow access to the non-volatile memory only by the microcontroller before power-off, wherein the device interface is coupled to the non-volatile memory and controls access to the non-volatile memory;

queuing one or more writes or other exchanges with the non-volatile memory; and

completing the one or more writes or other exchanges with the non-volatile memory before final power-off.

194. The method of claim 17, wherein the system further includes one or more additional processors, each coupled to the microcontroller and the non-volatile memory, and wherein the method further includes controlling the power or execution states of the processor and the one or more additional processors for sharing the non-volatile memory between the plurality of processors.

195. The method of claim 17, wherein the system further includes one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, and wherein the method further includes each additional microcontroller sharing the non-volatile memory with the processor.

196. The method of claim 17, wherein the system further includes one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, and wherein the method further includes:

each additional microcontroller sharing the non-volatile memory; and

each microcontroller suspending or preventing access to the non-volatile memory by other microcontrollers and the processor while it accesses the non-volatile memory.

197. The method of claim 17, further comprising:

intercepting communications between a system interface and the non-volatile memory, wherein the system interface is coupled to the processor, and wherein the microcontroller is situated between the system interface and the non-volatile memory; and

a device interface directing or restricting access to the non-volatile memory by the processor or the microcontroller, wherein the device interface is included in or associated with the non-volatile memory.

198. The method of claim 197, wherein the microcontroller is configured to set one or more control signals to configure or redirect the device interface to grant exclusive access to the non-volatile memory to the microcontroller.

199. The method of claim 17, wherein the non-volatile memory comprises flash memory.

200. The method of claim 17, wherein said fetching program instructions or data from the non-volatile memory further comprises fetching initialization code or data for the system.

14

201. The method of claim 200, further comprising:

fetching program instructions or data related to pre-boot security operations; and

performing the pre-boot security operations prior to boot-up of the system.

202. The method of claim 17, wherein fetching the program instructions or data from the non-volatile memory occurs via a non-volatile memory bus that couples the microcontroller and the non-volatile memory.

203. The method of claim 202, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the method further comprises the microcontroller switchably intercepting or passing communications between the processor and the non-volatile memory.

204. The method of claim 203, wherein the host memory bus and the non-volatile memory bus are SPI (serial peripheral interface) buses.

205. The method of claim 203, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, and wherein the microcontroller switchably intercepting or passing communications comprises the microcontroller controlling the switch to switchably intercept or pass communications between the processor and the non-volatile memory.

206. The method of claim 205, wherein the switch comprises a Q switch or a conceptual Q switch.

207. The method of claim 205, wherein switchably intercepting communications between the processor and the non-volatile memory comprises placing the switch into a high-impedance state.

208. The method of claim 203, wherein the host memory bus and the non-volatile memory bus each comprises:

a data line;

a clock line; and

a CSb (Chip Select bar) line.

209. The method of claim 203, wherein the host memory bus and the non-volatile memory bus each comprises a plurality of lines.

210. The method of claim 209, wherein each line of the host memory bus and the non-volatile memory bus is independently switchable.

211. The method of claim 209, further comprising:

switching the lines of the host memory bus and the non-volatile memory bus together to suspend or inhibit communications between a system interface and the non-volatile memory, wherein the system interface is coupled to the processor, and further coupled to the microcontroller for communicating signals between the system interface and the microcontroller.

212. The method of claim 209, wherein the lines of the host memory bus and the non-volatile memory bus are coupled to respective buffers for communicating data or signals between the non-volatile memory bus and the microcontroller.

213. The method of claim 17, further comprising:

communicating management signals between a system interface and the microcontroller, wherein the system interface is coupled to the processor and is further coupled to the microcontroller via a system management bus, and wherein the microcontroller comprises a system reset line for communicating the system reset signal or other signal to or from the system interface.

\* \* \* \* \*



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| APPLICATION NO.  | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|--|-------------|----------------------|---------------------|------------------|
| 90/011,754   | 06/21/2011  | 7,930,576 B2         | JCLA38862-RE        | 4071             |
| 35690  | 7590        | 06/21/2012           | EXAMINER            |                  |
| MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.<br>P.O. BOX 398<br>AUSTIN, TX 78767-0398 |             |                      | ART UNIT            | PAPER NUMBER     |

DATE MAILED: 06/21/2012

Please find below and/or attached an Office communication concerning this application or proceeding.



UNITED STATES PATENT AND TRADEMARK OFFICE

---

Commissioner for Patents  
United States Patents and Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS  
JIAWEI HUANG (J.C. PATENTS)  
4, VENTURE  
SUITE 250  
IRVINE, CA 92618

Date:

**MAILED**

**JUN 21 2012**

**CENTRAL REEXAMINATION UNIT**

**EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM**

REEXAMINATION CONTROL NO. : 90011754  
PATENT NO. : 7930576  
ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified ex parte reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the ex parte reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

---

|   |                |                            |
|---|----------------|----------------------------|
| <b>Notice of Intent to Issue<br/>Ex Parte Reexamination Certificate</b> | Control No.    | Patent Under Reexamination |
|   | 90/011,754     | 7,930,576 B2 ET            |
|   | Examiner       | Art Unit                   |
|   | BEHZAD PEIKARI | 3992                       |

**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --**

1. ☒ Prosecution on the merits is (or remains) closed in this *ex parte* reexamination proceeding. This proceeding is subject to reopening at the initiative of the Office or upon petition. *Cf.* 37 CFR 1.313(a). A Certificate will be issued in view of
  - (a) ☒ Patent owner's communication(s) filed: 23 April 2012.
  - (b) ☐ Patent owner's failure to file an appropriate timely response to the Office action mailed: \_\_\_\_\_.
  - (c) ☐ Patent owner's failure to timely file an Appeal Brief (37 CFR 41.31).
  - (d) ☐ The decision on appeal by the ☐ Board of Patent Appeals and Interferences ☐ Court dated \_\_\_\_\_
  - (e) ☐ Other: \_\_\_\_\_.
2. The Reexamination Certificate will indicate the following:
  - (a) Change in the Specification: ☐ Yes ☒ No
  - (b) Change in the Drawing(s): ☐ Yes ☒ No
  - (c) Status of the Claim(s):
    - (1) Patent claim(s) confirmed: 1-23.
    - (2) Patent claim(s) amended (including dependent on amended claim(s)): \_\_\_\_\_
    - (3) Patent claim(s) canceled: \_\_\_\_\_.
    - (4) Newly presented claim(s) patentable: 24-213.
    - (5) Newly presented canceled claims: \_\_\_\_\_.
    - (6) Patent claim(s) ☐ previously ☐ currently disclaimed: \_\_\_\_\_
    - (7) Patent claim(s) not subject to reexamination: \_\_\_\_\_.
3. ☒ Note the attached statement of reasons for patentability and/or confirmation. Any comments considered necessary by patent owner regarding reasons for patentability and/or confirmation must be submitted promptly to avoid processing delays. Such submission(s) should be labeled: "Comments On Statement of Reasons for Patentability and/or Confirmation."
4. ☐ Note attached NOTICE OF REFERENCES CITED (PTO-892).
5. ☐ Note attached LIST OF REFERENCES CITED (PTO/SB/08 or PTO/SB/08 substitute).
6. ☐ The drawing correction request filed on \_\_\_\_\_ is: ☐ approved ☐ disapproved.
7. ☐ Acknowledgment is made of the priority claim under 35 U.S.C. § 119(a)-(d) or (f).
  - a) ☐ All b) ☐ Some\* c) ☐ None of the certified copies have
    - ☐ been received.
    - ☐ not been received.
    - ☐ been filed in Application No. \_\_\_\_\_.
    - ☐ been filed in reexamination Control No. \_\_\_\_\_.
    - ☐ been received by the International Bureau in PCT Application No. \_\_\_\_\_.

\* Certified copies not received: \_\_\_\_\_.
8. ☐ Note attached Examiner's Amendment.
9. ☐ Note attached Interview Summary (PTO-474).
10. ☐ Other: \_\_\_\_\_.

**All correspondence** relating to this reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of this Office action.

cc: Requester (if third party requester)

U.S. Patent and Trademark Office  
PTOL-469 (Rev. 07-10)

Notice of Intent to Issue Ex Parte Reexamination Certificate

Part of Paper No 20120617

## **STATEMENT OF REASONS FOR PATENTABILITY AND/OR CONFIRMATION**

The following is an examiner's statement of reasons for patentability and/or confirmation of the claims found patentable in this reexamination proceeding:

In the non-final Office action mailed February 23, 2012, claims 1-2, 4-7 and 9-10 were rejected under 35 U.S.C. § 102(e) as being anticipated by DeRoo and claims 3, 8 and 11-23 were rejected under 35 U.S.C. § 103(a) as being unpatentable over DeRoo.

Patent owner filed a response on April 23, 2012. Some of the arguments included in that response are deemed convincing:

(1) With regard to claims 1-10, the arguments presented as section "2.", spanning pages 48-52, are deemed convincing to demonstrate that claims 1-2, 4-7 and 9-10 were not anticipated by DeRoo and claims 3 and 8 were not obvious over DeRoo.

Specifically, patent owner has demonstrated that in the system of DeRoo the access of SCP 706 to common memory device 704 is not certain or guaranteed to be free from interruption or interference by CPU 702. Thus, CPU 702 can preempt the microcontroller SCP 706 at any time, which teaches away from the claimed "assured safe access" of a microcontroller to a memory.

(2) With regard to claims 11-23, the examiner argued that "it would have been obvious to one having ordinary skill in the art at the time the invention was made to hold the processor in the system of DeRoo in a reset state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller exclusive access to the non-volatile memory until the reset signal is released, since (1) DeRoo specifically taught the use of a reset

Art Unit: 3992

state to prevent an element from accessing the non-volatile memory; namely, holding the microcontroller in a reset state to prohibit the microcontroller from accessing the non-volatile memory, while allowing the processor to do so, (2) DeRoo specifically taught the use of a state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller exclusive access to the non-volatile memory; namely, holding a processor in a wait state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so and (3) it may have been advantageous to take make use of any idle time during the wait state of the processor by performing a reset operation in the processor to run new updates, etc.”

However, patent owner’s arguments presented as section “2.”, spanning pages 66-67, are deemed convincing to demonstrate that claims 11-23 were not obvious over DeRoo.

Specifically, patent owner has demonstrated DeRoo fails to disclose the claimed “prohibiting the processor from accessing the non-volatile memory”. In rejecting claim 11, the previous rejection stated that DeRoo discloses “holding a processor in a wait state to prohibit the processor from accessing the non-volatile memory.”

However, as explained above, CPU 702 is not prohibited from accessing common memory device 704 when SCP 706 has exclusive access, because even “when the SCP has exclusive access of the common memory device 704, the CPU 702 can gain control by forcing the SCP 706 into reset” (*see DeRoo, column 81, lines 64-66*). That is, even when SCP 706 has exclusive access, CPU 702 may unconditionally regain control over common memory device 704.

Art Unit: 3992

Consequently, the previous rejections are withdrawn and claims 1-23 are confirmed.

Furthermore, the new claims 24-213 are dependent, either directly or indirectly, upon the original patent claims and are deemed patentable.

Art Unit: 3992

All correspondence relating to this *ex parte* reexamination proceeding should be directed:

By Mail to: Mail Stop *Ex Parte* Reexam  
Central Reexamination Unit  
Commissioner for Patents  
United States Patent & Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450

By FAX to: (571) 273-9900  
Central Reexamination Unit

By hand to: Customer Service Window  
Randolph Building  
401 Dulany Street  
Alexandria, VA 22314

Registered users of EFS-Web may alternatively submit such correspondence via the electronic filing system EFS-Web, at <https://efs.uspto.gov/efile/myportal/efs-registered>. EFS-Web offers the benefit of quick submission to the particular area of the Office that needs to act on the correspondence. Also, EFS-Web submissions are "soft scanned" (i.e., electronically uploaded) directly into the official file for the reexamination proceeding, which offers parties the opportunity to review the content of their submissions after the "soft scanning" process is complete.

Any inquiry concerning this communication should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

Signed:

Conferees: /r.g.f./


**/B. James Peikari/**

B. James Peikari  
Primary Examiner  
Art Unit 3992

Alexander Kosowski  
Supervisor  
Art Unit 3992






|   |                         |   |
|---|-------------------------|---|
| <b>Reexamination</b><br> | Application/Control No. | Applicant(s)/Patent Under Reexamination |
|   | 90/011,754              | 7,930,576 B2 ET AL.                     |
|   | Certificate Date        | Certificate Number                      |
|   |                         | C1                                      |

|   |                         |                                       |   |
|---|-------------------------|---------------------------------------|---|
| Requester   | Correspondence Address: | <input type="checkbox"/> Patent Owner | <input checked="" type="checkbox"/> Third Party |
| <p>MEYERTONS, HOOD, KIVLIN, KOWERT &amp; GOETZEL, P.C.<br/> P.O. BOX 398<br/> AUSTIN, TX 78767-0398</p> |                         |                                       |   |

|   |                            |                   |
|---|----------------------------|-------------------|
| LITIGATION REVIEW <input checked="" type="checkbox"/> | BJP<br>(examiner initials) | 6/16/12<br>(date) |
| Case Name   | Director Initials          |                   |
| None  | ASX for dy                 |                   |
|   |                            |                   |
|   |                            |                   |
|   |                            |                   |
|   |                            |                   |

| COPENDING OFFICE PROCEEDINGS |        |
|------------------------------|--------|
| TYPE OF PROCEEDING           | NUMBER |
| 1. None                      |        |
| 2.                           |        |
| 3.                           |        |
| 4.                           |        |

|  |  |   |
|--|--|---|
| <b>Issue Classification</b><br> | <b>Application/Control No.</b><br>90/011,754 | <b>Applicant(s)/Patent under Reexamination</b><br>7,930,576 B2 ET AL. |
|  | <b>Examiner</b><br>BEHZAD PEIKARI            | <b>Art Unit</b><br>3992   |

| ISSUE CLASSIFICATION |                                   |   |          |  |  |                              |   |   |             |      |   |   |
|----------------------|-----------------------------------|---|----------|--|--|------------------------------|---|---|-------------|------|---|---|
| ORIGINAL             |                                   |   |          |  |  | INTERNATIONAL CLASSIFICATION |   |   |             |      |   |   |
| CLASS                |                                   |   | SUBCLASS |  |  | CLAIMED                      |   |   | NON-CLAIMED |      |   |   |
| 713                  |                                   |   | 324      |  |  | G                            | 6 | F | 1           | /26  |   | / |
| CROSS REFERENCES     |                                   |   |          |  |  | G                            | 6 | F | 15          | /177 |   | / |
| CLASS                | SUBCLASS (ONE SUBCLASS PER BLOCK) |   |          |  |  |                              |   |   |             |      |   |   |
| 713                  | 100                               | 2 |          |  |  |                              |   |   |             |      | / |   |
|                      |                                   |   |          |  |  |                              |   |   |             |      | / |   |
|                      |                                   |   |          |  |  |                              |   |   |             |      | / |   |
|                      |                                   |   |          |  |  |                              |   |   |             |      | / |   |
|                      |                                   |   |          |  |  |                              |   |   |             |      | / |   |

|                                      |   |  |
|--------------------------------------|---|--|
| -----<br>(Assistant Examiner) (Date) | /B. James Peikari/<br><br>B. James Peikari 6/16/12<br>(Primary Examiner) (Date) | <b>Total Claims Allowed: 213</b><br><br>O.G. Print Claim(s) 1<br>O.G. Print Fig. 4 |
| (Legal Instruments Examiner) (Date)  |   |  |

| <input checked="" type="checkbox"/> Claims renumbered in the same order as presented by applicant |          |       |          |       |          |       |          |       |          |       |          | <input type="checkbox"/> CPA |          | <input type="checkbox"/> T.D. |          | <input type="checkbox"/> R.1.47 |  |
|---|----------|-------|----------|-------|----------|-------|----------|-------|----------|-------|----------|------------------------------|----------|-------------------------------|----------|---------------------------------|--|
| Final   | Original | Final | Original | Final | Original | Final | Original | Final | Original | Final | Original | Final                        | Original | Final                         | Original |                                 |  |
|   | 1        |       | 31       |       | 61       |       | 91       |       | 121      |       | 151      |                              | 181      |                               |          |                                 |  |
|   | 2        |       | 32       |       | 62       |       | 92       |       | 122      |       | 152      |                              | 182      |                               |          |                                 |  |
|   | 3        |       | 33       |       | 63       |       | 93       |       | 123      |       | 153      |                              | 183      |                               |          |                                 |  |
|   | 4        |       | 34       |       | 64       |       | 94       |       | 124      |       | 154      |                              | 184      |                               |          |                                 |  |
|   | 5        |       | 35       |       | 65       |       | 95       |       | 125      |       | 155      |                              | 185      |                               |          |                                 |  |
|   | 6        |       | 36       |       | 66       |       | 96       |       | 126      |       | 156      |                              | 186      |                               |          |                                 |  |
|   | 7        |       | 37       |       | 67       |       | 97       |       | 127      |       | 157      |                              | 187      |                               |          |                                 |  |
|   | 8        |       | 38       |       | 68       |       | 98       |       | 128      |       | 158      |                              | 188      |                               |          |                                 |  |
|   | 9        |       | 39       |       | 69       |       | 99       |       | 129      |       | 159      |                              | 189      |                               |          |                                 |  |
|   | 10       |       | 40       |       | 70       |       | 100      |       | 130      |       | 160      |                              | 190      |                               |          |                                 |  |
|   | 11       |       | 41       |       | 71       |       | 101      |       | 131      |       | 161      |                              | 191      |                               |          |                                 |  |
|   | 12       |       | 42       |       | 72       |       | 102      |       | 132      |       | 162      |                              | 192      |                               |          |                                 |  |
|   | 13       |       | 43       |       | 73       |       | 103      |       | 133      |       | 163      |                              | 193      |                               |          |                                 |  |
|   | 14       |       | 44       |       | 74       |       | 104      |       | 134      |       | 164      |                              | 194      |                               |          |                                 |  |
|   | 15       |       | 45       |       | 75       |       | 105      |       | 135      |       | 165      |                              | 195      |                               |          |                                 |  |
|   | 16       |       | 46       |       | 76       |       | 106      |       | 136      |       | 166      |                              | 196      |                               |          |                                 |  |
|   | 17       |       | 47       |       | 77       |       | 107      |       | 137      |       | 167      |                              | 197      |                               |          |                                 |  |
|   | 18       |       | 48       |       | 78       |       | 108      |       | 138      |       | 168      |                              | 198      |                               |          |                                 |  |
|   | 19       |       | 49       |       | 79       |       | 109      |       | 139      |       | 169      |                              | 199      |                               |          |                                 |  |
|   | 20       |       | 50       |       | 80       |       | 110      |       | 140      |       | 170      |                              | 200      |                               |          |                                 |  |
|   | 21       |       | 51       |       | 81       |       | 111      |       | 141      |       | 171      |                              | 201      |                               |          |                                 |  |
|   | 22       |       | 52       |       | 82       |       | 112      |       | 142      |       | 172      |                              | 202      |                               |          |                                 |  |
|   | 23       |       | 53       |       | 83       |       | 113      |       | 143      |       | 173      |                              | 203      |                               |          |                                 |  |
|   | 24       |       | 54       |       | 84       |       | 114      |       | 144      |       | 174      |                              | 204      |                               |          |                                 |  |
|   | 25       |       | 55       |       | 85       |       | 115      |       | 145      |       | 175      |                              | 205      |                               |          |                                 |  |
|   | 26       |       | 56       |       | 86       |       | 116      |       | 146      |       | 176      |                              | 206      |                               |          |                                 |  |
|   | 27       |       | 57       |       | 87       |       | 117      |       | 147      |       | 177      |                              | 207      |                               |          |                                 |  |
|   | 28       |       | 58       |       | 88       |       | 118      |       | 148      |       | 178      |                              | 208      |                               |          |                                 |  |
|   | 29       |       | 59       |       | 89       |       | 119      |       | 149      |       | 179      |                              | 209      |                               |          |                                 |  |
|   | 30       |       | 60       |       | 90       |       | 120      |       | 150      |       | 180      |                              | 210      |                               |          |                                 |  |

**Search Notes**

Application/Control No.

90/011,754

Examiner

BEHZAD PEIKARI

Applicant(s)/Patent under  
Reexamination

7,930,576 B2 ET AL.

Art Unit

3992

**SEARCHED**

| Class | Subclass | Date | Examiner |
|-------|----------|------|----------|
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |

**INTERFERENCE SEARCHED**

| Class | Subclass | Date | Examiner |
|-------|----------|------|----------|
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |
|       |          |      |          |

**SEARCH NOTES  
(INCLUDING SEARCH STRATEGY)**

|   | DATE      | EXMR |
|---|-----------|------|
| Review of the prosecution history of U.S. Patent No. 7,930,576.         | 8/24/2011 | BJP  |
| Updated review of the prosecution history of U.S. Patent No. 7,930,576. | 6/16/12   | BJP  |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |



## UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
 United States Patent and Trademark Office  
 Address: COMMISSIONER FOR PATENTS  
 P.O. Box 1450  
 Alexandria, Virginia 22313-1450  
 www.uspto.gov

## BIB DATA SHEET

CONFIRMATION NO. 4071

|  |   |  |                               |  |                           |                                |
|--|---|--|-------------------------------|--|---------------------------|--------------------------------|
| <b>SERIAL NUMBER</b><br>90/011,754   | <b>FILING or 371(c)<br/>DATE</b><br>06/21/2011<br><b>RULE</b>   | <b>CLASS</b><br>713                                      | <b>GROUP ART UNIT</b><br>3992 | <b>ATTORNEY DOCKET NO.</b><br>JCLA38862-RE                   |                           |                                |
| <b>APPLICANTS</b><br>7,930,576 B2, Residence Not Provided;<br>STANDARD MICROSYSTEMS CORPORATION (OWNER), HAUPPAUGE, NY;<br>FELIX YEN (3RD PTY. REQ.), TAIPEI, TAIWAN;<br>JIAWEI HUANG (J.C. PATENTS), IRVINE, CA<br><b>** CONTINUING DATA *****</b><br>This application is a REX of 11/958,601 12/18/2007 PAT 7,930,576<br>which claims benefit of 60/910,863 04/10/2007<br><b>** FOREIGN APPLICATIONS *****</b><br><b>** IF REQUIRED, FOREIGN FILING LICENSE GRANTED **</b> |   |  |                               |  |                           |                                |
| Foreign Priority claimed <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No<br>35 USC 119(a-d) conditions met <input type="checkbox"/> Yes <input checked="" type="checkbox"/> No<br>Verified and Acknowledged <u>/BEHZAD PEIKARI/</u><br>Examiner's Signature  |   | <input type="checkbox"/> Met after Allowance<br>Initials | <b>STATE OR COUNTRY</b>       | <b>SHEETS DRAWINGS</b>                                       | <b>TOTAL CLAIMS</b><br>23 | <b>INDEPENDENT CLAIMS</b><br>4 |
| <b>ADDRESS</b><br>MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.<br>P.O. BOX 398<br>AUSTIN, TX 78767-0398<br>UNITED STATES  |   |  |                               |  |                           |                                |
| <b>TITLE</b><br>SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM   |   |  |                               |  |                           |                                |
| <b>FILING FEE RECEIVED</b><br>2520   | FEES: Authority has been given in Paper<br>No. _____ to charge/credit DEPOSIT ACCOUNT<br>No. _____ for following: |  |                               | <input type="checkbox"/> All Fees                            |                           |                                |
|  |   |  |                               | <input type="checkbox"/> 1.16 Fees (Filing)                  |                           |                                |
|  |   |  |                               | <input type="checkbox"/> 1.17 Fees (Processing Ext. of time) |                           |                                |
|  |   |  |                               | <input type="checkbox"/> 1.18 Fees (Issue)                   |                           |                                |
|  |   |  |                               | <input type="checkbox"/> Other _____                         |                           |                                |
|  |   |  |                               | <input type="checkbox"/> Credit                              |                           |                                |

## EX PARTE REEXAMINATION

Reexamination  
Control. No.: 90/011,754  
Inventor: Ian F. Harris, et al.  
Request Filing Date: June 21, 2011  
U.S. Patent No.: 7,930,576  
Title: SHARING NON-SHARABLE  
DEVICES BETWEEN AN  
EMBEDDED CONTROLLER  
AND A PROCESSOR IN A  
COMPUTER SYSTEM

Conf. No.: 4071

On: April 23, 2012 / Anthony M. Petro /  
Date Anthony M. Petro, #59,391

## Page 25 of 264

## **TABLE OF CONTENTS**

|  |    |
|--|----|
| IN THE CLAIMS: .....   | 4  |
| REMARKS .....  | 33 |
| I. Introduction .....  | 34 |
| II. Overview of '576 Patent .....  | 36 |
| III. Overview of the DeRoo Reference.....  | 39 |
| IV. Response to Rejections Based on DeRoo .....  | 45 |
| A. Claim 1: .....  | 45 |
| 1. The combination of DeRoo's SCP 706 and HUI 700 does not correspond to the recited "microcontroller." .....  | 46 |
| 2. DeRoo fails to disclose "a first state wherein the microcontroller is assured safe access to the non-volatile memory." .....  | 48 |
| a. For a microcontroller to be "assured safe access" to a memory, under a broadest reasonable interpretation of that term, the microcontroller's access to the memory cannot be arbitrarily preempted by a processor. .... | 48 |
| b. In DeRoo's system, CPU 702 can arbitrarily preempt SCP 706 at any time, which is precisely the opposite of what is needed for a microcontroller to be "assured safe access" to a memory. ....                           | 50 |
| 3. DeRoo fails to disclose that a microcontroller is configured to "hold the system in the first state." .....   | 52 |
| 4. DeRoo fails to disclose that a microcontroller is configured to "release the system from the first state." .....  | 54 |
| 5. Contrary to the Office Action's position, DeRoo does not disclose that a microcontroller is configured to perform certain actions <i>in response to</i> a change in state of the CPU SYSTEM ACCESS signal. ....         | 55 |
| 6. Because DeRoo fails to disclose or suggest numerous features of claim 1, DeRoo fails to support the rejection of claim 1. ....  | 56 |
| B. Claim 2: .....  | 56 |
| C. Claim 4: .....  | 57 |
| D. Claim 5: .....  | 57 |

|  |    |
|--|----|
| E. Claim 6:.....   | 58 |
| F. Claim 7:.....   | 59 |
| G. Claim 9: .....  | 59 |
| H. Claim 10: .....   | 59 |
| I. Claim 3:.....   | 59 |
| J. Claim 8: .....  | 60 |
| K. Claim 11: .....   | 60 |
| 1. The proposed modification would impermissibly change a fundamental principle of operation of DeRoo and would impermissibly render DeRoo unsatisfactory for its intended purpose.....          | 62 |
| 2. Contrary to the Office Action’s assertion, DeRoo fails to disclose “prohibiting the processor from accessing the non-volatile memory,” leaving some features of claim 11 unaccounted for..... | 66 |
| L. Claim 12:.....  | 67 |
| M. Claim 13: .....   | 67 |
| N. Claim 14: .....   | 67 |
| O. Claim 15: .....   | 68 |
| P. Claim 16:.....  | 68 |
| Q. Claim 17: .....   | 68 |
| R. Claim 18: .....   | 69 |
| S. Claim 19:.....  | 69 |
| T. Claim 20:.....  | 69 |
| U. Claim 21: .....   | 70 |
| V. Claim 22: .....   | 70 |
| W. Claim 23: .....   | 70 |
| V. New Dependent Claims .....  | 71 |
| VI. Conclusion.....  | 72 |

**IN THE CLAIMS:**

Please add the following claims, which are shown in accordance with 37 C.F.R. § 1.530:

24. The system of claim 1, wherein the change in system state to the first state is performed by hardware.

25. The system of claim 1, wherein the change in system state to the first state is in response to a hardware signal.

26. The system of claim 1, wherein the change in system state to the first state is in response to a software event.

27. The system of claim 1, wherein the change in system state to the first state is in response to a token.

28. The system of claim 1, wherein the change in system state to the first state is in response to the system passing the microcontroller a token which grants the microcontroller ownership of the memory through a software handshake.

29. The system of claim 1, wherein the change in system state to the first state is in response to a software condition.

30. The system of claim 1, wherein the change in system state to the first state is in response to a software message.

31. The system of claim 1, wherein the change in system state to the first state is in response to a software handshake.

32. The system of claim 1, wherein the first state comprises a power on reset (POR) state.



33. The system of claim 1, wherein the first state comprises a system reset state.

34. The system of claim 1, wherein the first state comprises a sleep state.

35. The system of claim 1, wherein to hold the system in the first state, the microcontroller is configured to:

in response to a power on reset, hold a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory.

36. The system of claim 35, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller while the system is held in the first state, the microcontroller is configured to:

fetch the program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller while the system reset signal is held in the reset state.

37. The system of claim 35, wherein to switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state after the program instructions or data have been loaded, the microcontroller is configured to:

after the program instructions or data have been loaded, permit the processor in the system access to the non-volatile memory, and release the system reset signal from the reset state.

38. The system of claim 1, wherein the first state comprises a state wherein it is known by the microcontroller that the system cannot or will not access the device.

39. The system of claim 1, wherein the microcontroller is included within a system-on-chip (SoC).

40. The system of claim 1, wherein the microcontroller comprises a keyboard controller for

managing or controlling communications between a keyboard and the processor.

41. The system of claim 40, wherein the non-volatile memory stores keyboard controller (KBC) code.

42. The system of claim 1, wherein the program instructions or data are usable by the microcontroller for startup, general, or reset functionality.

43. The system of claim 1, wherein the program instructions or data are usable by the microcontroller for initializing the microcontroller.

44. The system of claim 1, wherein during system shutdown, the microcontroller is further configured to:

configure the non-volatile memory to allow access to the non-volatile memory only by the microcontroller before power-off;  
queue one or more writes or other exchanges with the non-volatile memory; and  
complete the one or more writes or other exchanges with the non-volatile memory before final power-off.

45. The system of claim 1, further comprising:

a device interface coupled to the non-volatile memory, wherein the device interface controls access to the non-volatile memory, wherein during system shutdown, the microcontroller is further configured to:

redirect the device interface to allow access to the non-volatile memory only by the microcontroller before power-off;  
queue one or more writes or other exchanges with the non-volatile memory; and  
complete the one or more writes or other exchanges with the non-volatile memory before final power-off.

46. The system of claim 1, further comprising:

one or more additional processors, each coupled to the microcontroller and the non-volatile memory, and wherein the microcontroller has control over the power or execution states of the processor and the one or more additional processors for sharing the non-volatile memory between the plurality of processors.

47. The system of claim 1, further comprising:

one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, wherein each additional microcontroller is configured to share the non-volatile memory with the processor.

48. The system of claim 1, further comprising:

one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, wherein each additional microcontroller is configured to share the non-volatile memory, and wherein each microcontroller is configured to suspend or prevent access to the non-volatile memory by other microcontrollers and the processor while it accesses the non-volatile memory.

49. The system of claim 1, further comprising:

a system interface, coupled to the processor, wherein the microcontroller is situated between the system interface and the non-volatile memory, and wherein the microcontroller is configured to intercept communications between the system interface and the non-volatile memory; and

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured to direct or restrict access to the non-volatile memory by the processor or the microcontroller.

50. The system of claim 49, wherein the microcontroller is configured to set one or more control signals to configure or redirect the device interface to grant exclusive access to the non-volatile memory to the microcontroller.

51. The system of claim 1, wherein the non-volatile memory comprises flash memory.

52. The system of claim 1, wherein the change in system state to the first state is in response to a management signal, wherein to hold the system in the first state, the microcontroller is further configured to hold the management signal in a corresponding state, or to hold or assert another management signal in a specified state.

53. The system of claim 52, further comprising:

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured by the microcontroller to direct or restrict access to the non-volatile memory by the processor or the microcontroller independent of the management signal.

54. The system of claim 52, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller via a system management bus configured for communicating management signals between the system interface and the microcontroller.

55. The system of claim 52, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller via a system interface bus configured for communicating management signals between the system interface and the microcontroller.

56. The system of claim 52, wherein the management signal or the another management signal is invoked by the microcontroller.

57. The system of claim 52, wherein the management signal or the another management signal is not invoked by the microcontroller.

58. The system of claim 52, wherein the management signal or the another management signal is a software signal.

59. The system of claim 52, wherein the management signal or the other management signal is a hardware signal.

60. The system of claim 1, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via software.

61. The system of claim 1, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via hardware.

62. The system of claim 1, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via a combination of software and hardware.

63. The system of claim 1, wherein the change in system state to the first state is not dependent on a management signal.

64. The system of claim 1, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller, the microcontroller is configured to:

fetch initialization code or data for the system.

65. The system of claim 64, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller, the microcontroller is further configured to:

fetch program instructions or data related to pre-boot security operations; and  
perform the pre-boot security operations prior to boot-up of the system.

66. The system of claim 1, wherein the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus.

67. The system of claim 66, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable to switchably intercept or pass communications between the processor and the non-volatile memory.

68. The system of claim 67, wherein the host memory bus and the non-volatile memory bus are SPI (serial peripheral interface) buses.

69. The system of claim 67, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor and the non-volatile memory.

70. The system of claim 69, wherein the switch comprises a Q switch or a conceptual Q switch.

71. The system of claim 69, wherein to switchably intercept or pass communications between the processor and the non-volatile memory, the microcontroller is configured to place the switch into a high-impedance state.

72. The system of claim 69, wherein to switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state, the microcontroller is configured to:

reconfigure the access to the non-volatile memory via the switch, and release the system from the first state, thereby permitting the processor access to the non-volatile memory.

73. The system of claim 69,

wherein the first state comprises a reset state, wherein to hold the system in the first state,

the microcontroller is configured to maintain assertion of a system reset signal; and  
wherein to switch access to the non-volatile memory from the microcontroller to the  
processor, and release the system from the first state, the microcontroller is configured to:  
terminate a prior suspension or preemption of communications between the host  
interface and the non-volatile memory via the switch, thereby allowing resumption of the  
communications; and  
de-assert the system reset signal.

74. The system of claim 67, wherein the host memory bus and the non-volatile memory bus  
each comprises:

- a data line;
- a clock line; and
- a CSb (Chip Select bar) line.

75. The system of claim 67, wherein the host memory bus and the non-volatile memory bus  
each comprises a plurality of lines.

76. The system of claim 75, wherein each line of the host memory bus and the non-volatile  
memory bus is independently switchable.

77. The system of claim 75, further comprising:  
a system interface, coupled to the processor, and further coupled to the microcontroller  
for communicating signals between the system interface and the microcontroller;  
wherein the lines of the host memory bus and the non-volatile memory bus are switchable  
together to suspend or inhibit communications between the system interface and the non-volatile  
memory.

78. The system of claim 75, wherein the lines of the host memory bus and the non-volatile  
memory bus are coupled to respective buffers for communicating data or signals between the  
non-volatile memory bus and the microcontroller.

79. The system of claim 78, further comprising:

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured by the microcontroller to:

when the system is held in the first state, preclude the processor from accessing the non-volatile memory.

80. The system of claim 79, wherein the microcontroller is further configured to:

once the microcontroller has completed fetching program instructions or data from the non-volatile memory, or has determined that it is no longer assured of safe access to the non-volatile memory, reconfigure the device interface via control signals to enable access by the processor to the non-volatile memory.

81. The system of claim 1,

wherein the first state comprises a reset state, wherein to hold the system in the first state, the microcontroller is configured to maintain assertion of a system reset signal; and

wherein to switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state, the microcontroller is configured to:

terminate a prior suspension or preemption of communications between the processor and the non-volatile memory, thereby allowing communications to occur between the processor and the non-volatile memory; and

de-assert the system reset signal.

82. The system of claim 1, further comprising:

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured by the microcontroller to:

when the system is held in the first state, preclude the processor from accessing the non-volatile memory.

83. The system of claim 1, wherein the microcontroller is further configured to:



while the system is held in the first state, exchange data or program instructions with the non-volatile memory.

84. The system of claim 83, wherein to exchange data or program instructions with the non-volatile memory, the microcontroller is configured to:

invoke a read or write by another component of the system.

85. The system of claim 1, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller via a system management bus configured for communicating management signals between the system interface and the microcontroller;

wherein the microcontroller comprises a system reset line for communicating a reset signal or other signal to or from the system interface.

86. The system of claim 85, wherein the reset signal or other signal corresponds with or invokes a suspension of non-volatile memory access by the processor.

87. The system of claim 85, wherein the reset signal or other signal indicates a system state in which the microcontroller has safe access to the non-volatile memory.

88. The system of claim 85, wherein to hold the system in the first state, the microcontroller is configured to hold the system reset signal or other signal in a reset state.

89. The method of claim 6, wherein the change in system state to the first state is performed by hardware.

90. The method of claim 6, wherein the change in system state to the first state is in response to a hardware signal.

91. The method of claim 6, wherein the change in system state to the first state is in response to a

software event.

92. The method of claim 6, wherein the change in system state to the first state is in response to a token.

93. The method of claim 6, wherein the change in system state to the first state is in response to the system passing the microcontroller a token which grants the microcontroller ownership of the memory through a software handshake.

94. The method of claim 6, wherein the change in system state to the first state is in response to a software condition.

95. The method of claim 6, wherein the change in system state to the first state is in response to a software message.

96. The method of claim 6, wherein the change in system state to the first state is in response to a software handshake.

97. The method of claim 6, wherein the first state comprises a power on reset (POR) state.

98. The method of claim 6, wherein the first state comprises a system reset state.

99. The method of claim 6, wherein the first state comprises a sleep state.

100. The method of claim 6, wherein holding the system in the first state comprises, in response to a power on reset, holding a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory.

101. The method of claim 100, wherein said fetching program instructions or data from the non-volatile memory and loading the program instructions or data into a memory of the

microcontroller while the system is held in the first state occur while the system reset signal is held in the reset state.

102. The method of claim 100, wherein said switching access to the non-volatile memory from the microcontroller to the processor comprises permitting the processor in the system to access the non-volatile memory, and wherein releasing the system from the first state after the program instructions or data have been loaded comprises releasing the system reset signal from the reset state.

103. The method of claim 6, wherein the first state comprises a state wherein it is known by the microcontroller that the system cannot or will not access the device.

104. The method of claim 6, wherein the microcontroller is included within a system-on-chip (SoC).

105. The method of claim 6, wherein the microcontroller comprises a keyboard controller for managing or controlling communications between a keyboard and the processor.

106. The method of claim 105, wherein the non-volatile memory stores keyboard controller (KBC) code.

107. The method of claim 6, wherein the program instructions or data are usable by the microcontroller for startup, general, or reset functionality.

108. The method of claim 6, wherein the program instructions or data are usable by the microcontroller for initializing the microcontroller.

109. The method of claim 6, further comprising:  
during system shutdown:

configuring the non-volatile memory to allow access to the non-volatile memory only by the microcontroller before power-off;  
queuing one or more writes or other exchanges with the non-volatile memory; and  
completing the one or more writes or other exchanges with the non-volatile memory before final power-off.

110. The method of claim 6, further comprising:

during system shutdown:

redirecting a device interface to allow access to the non-volatile memory only by the microcontroller before power-off, wherein the device interface is coupled to the non-volatile memory and controls access to the non-volatile memory;  
queuing one or more writes or other exchanges with the non-volatile memory; and  
completing the one or more writes or other exchanges with the non-volatile memory before final power-off.

111. The method of claim 6, wherein the system further includes one or more additional processors, each coupled to the microcontroller and the non-volatile memory, and wherein the method further includes controlling the power or execution states of the processor and the one or more additional processors for sharing the non-volatile memory between the plurality of processors.

112. The method of claim 6, wherein the system further includes one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, and wherein the method further includes each additional microcontroller sharing the non-volatile memory with the processor.

113. The method of claim 6, wherein the system further includes one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, and wherein the method further includes:

each additional microcontroller sharing the non-volatile memory; and  
each microcontroller suspending or preventing access to the non-volatile memory by  
other microcontrollers and the processor while it accesses the non-volatile  
memory.

114. The method of claim 6, further comprising:

intercepting communications between a system interface and the non-volatile memory,  
wherein the system interface is coupled to the processor, and wherein the  
microcontroller is situated between the system interface and the non-volatile  
memory; and

a device interface directing or restricting access to the non-volatile memory by the  
processor or the microcontroller, wherein the device interface is included in or  
associated with the non-volatile memory.

115. The method of claim 114, wherein the microcontroller is configured to set one or more  
control signals to configure or redirect the device interface to grant exclusive access to the non-  
volatile memory to the microcontroller.

116. The method of claim 6, wherein the non-volatile memory comprises flash memory.

117. The method of claim 6, wherein the change in system state to the first state is in response to  
a management signal, wherein to hold the system in the first state, the microcontroller is further  
configured to hold the management signal in a corresponding state, or to hold or assert another  
management signal in a specified state.

118. The method of claim 117, further comprising:

a device interface directing or restricting access to the non-volatile memory by the  
processor or the microcontroller independent of the management signal, wherein the device  
interface is included in or associated with the non-volatile memory.

119. The method of claim 117, further comprising:

a system interface communicating management signals between the system interface and the microcontroller, wherein the system interface is coupled to the processor, and further coupled to the microcontroller via a system management bus.

120. The method of claim 117, further comprising:

a system interface communicating management signals between the system interface and the microcontroller, wherein the system interface is coupled to the processor, and further coupled to the microcontroller via a system interface bus.

121. The method of claim 117, wherein the management signal or the another management signal is invoked by the microcontroller.

122. The method of claim 117, wherein the management signal or the another management signal is not invoked by the microcontroller.

123. The method of claim 117, wherein the management signal or the another management signal is a software signal.

124. The method of claim 117, wherein the management signal or the other management signal is a hardware signal.

125. The method of claim 6, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via software.

126. The method of claim 6, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via hardware.

127. The method of claim 6, wherein switching access to the non-volatile memory from the processor to the microcontroller and from the microcontroller to the processor is performed via a combination of software and hardware.

128. The method of claim 6, wherein the change in system state to the first state is not dependent on a management signal.

129. The method of claim 6, wherein said fetching program instructions or data from the non-volatile memory further comprises fetching initialization code or data for the system.

130. The method of claim 129, further comprising:  
fetching program instructions or data related to pre-boot security operations; and  
performing the pre-boot security operations prior to boot-up of the system.

131. The method of claim 6, wherein fetching the program instructions or data from the non-volatile memory occurs via a non-volatile memory bus that couples the microcontroller and the non-volatile memory.

132. The method of claim 131, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the method further comprises the microcontroller switchably intercepting or passing communications between the processor and the non-volatile memory.

133. The method of claim 132, wherein the host memory bus and the non-volatile memory bus are SPI (serial peripheral interface) buses.

134. The method of claim 132, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, and wherein the microcontroller switchably intercepting or passing communications comprises the microcontroller controlling the switch to switchably intercept or pass communications between the processor and the non-volatile memory.

135. The method of claim 134, wherein the switch comprises a Q switch or a conceptual Q switch.

136. The method of claim 134, wherein switchably intercepting communications between the processor and the non-volatile memory comprises placing the switch into a high-impedance state.

137. The method of claim 134, wherein switching access to the non-volatile memory from the microcontroller to the processor comprises reconfiguring the access to the non-volatile memory via the switch.

138. The method of claim 134,  
wherein the first state comprises a reset state, wherein holding the system in the first state  
comprises maintaining assertion of a system reset signal; and  
wherein switching access to the non-volatile memory from the microcontroller to the  
processor comprises terminating a prior suspension or preemption of  
communications between the host interface and the non-volatile memory via the  
switch, thereby allowing resumption of the communications; and  
wherein releasing the system from the first state comprises de-asserting the system reset  
signal.

139. The method of claim 132, wherein the host memory bus and the non-volatile memory bus  
each comprises:

- a data line;
- a clock line; and
- a CSb (Chip Select bar) line.

140. The method of claim 132, wherein the host memory bus and the non-volatile memory bus  
each comprises a plurality of lines.



141. The method of claim 140, wherein each line of the host memory bus and the non-volatile memory bus is independently switchable.

142. The method of claim 140, further comprising:

switching the lines of the host memory bus and the non-volatile memory bus together to suspend or inhibit communications between a system interface and the non-volatile memory, wherein the system interface is coupled to the processor, and further coupled to the microcontroller for communicating signals between the system interface and the microcontroller.

143. The method of claim 140, wherein the lines of the host memory bus and the non-volatile memory bus are coupled to respective buffers for communicating data or signals between the non-volatile memory bus and the microcontroller.

144. The method of claim 143, further comprising:

when the system is held in the first state, precluding, by a device interface, the processor from accessing the non-volatile memory, wherein the device interface is included in or associated with the non-volatile memory.

145. The method of claim 140, further comprising:

once the microcontroller has completed fetching program instructions or data from the non-volatile memory, or has determined that it is no longer assured of safe access to the non-volatile memory, reconfiguring the device interface via control signals to enable access by the processor to the non-volatile memory.

146. The method of claim 6,

wherein the first state comprises a reset state, wherein holding the system in the first state comprises maintaining assertion of a system reset signal; and  
wherein switching access to the non-volatile memory from the microcontroller to the processor comprises terminating a prior suspension or preemption of

communications between the host interface and the non-volatile memory via the switch, thereby allowing resumption of the communications; and wherein releasing the system from the first state comprises de-asserting the system reset signal.

147. The method of claim 6, further comprising:

when the system is held in the first state, precluding, by a device interface, the processor from accessing the non-volatile memory, wherein the device interface is included in or associated with the non-volatile memory.

148. The method of claim 6, further comprising:

while the system is held in the first state, the microcontroller exchanging data or program instructions with the non-volatile memory.

149. The method of claim 148, wherein the microcontroller exchanging data or program instructions with the non-volatile memory comprises invoking a read or write by another component of the system.

150. The method of claim 6, further comprising:

communicating management signals between a system interface and the microcontroller, wherein the system interface is coupled to the processor and is further coupled to the microcontroller via a system management bus, and wherein the microcontroller comprises a system reset line for communicating a reset signal or other signal to or from the system interface.

151. The method of claim 150, wherein the reset signal or other signal corresponds with or invokes a suspension of non-volatile memory access by the processor.

152. The method of claim 150, wherein the reset signal or other signal indicates a system state in which the microcontroller has safe access to the non-volatile memory.

153. The system of claim 150, wherein holding the system in the first state comprises the microcontroller holding the system reset signal or other signal in a reset state.

154. The method of claim 6, wherein said holding the system in the first state is performed by the microcontroller.

155. The method of claim 6, wherein said switching access to the non-volatile memory from the processor to the microcontroller is performed by the microcontroller.

156. The method of claim 6, wherein said fetching program instructions or data from the non-volatile memory is performed by the microcontroller.

157. The method of claim 6, wherein said loading the program instructions or data into a memory of the microcontroller is performed by the microcontroller.

158. The method of claim 6, wherein said switching access to the non-volatile memory from the microcontroller to the processor is performed by the microcontroller.

159. The method of claim 6, wherein said releasing the system from the first state is performed by the microcontroller.

160. The system of claim 11, wherein the microcontroller is included within a system-on-chip (SoC).

161. The system of claim 11, wherein the microcontroller comprises a keyboard controller for managing or controlling communications between a keyboard and the processor.

162. The system of claim 161, wherein the non-volatile memory stores keyboard controller (KBC) code.

163. The system of claim 11, wherein the program instructions or data are usable by the microcontroller for startup, general, or reset functionality.

164. The system of claim 11, wherein the program instructions or data are usable by the microcontroller for initializing the microcontroller.

165. The system of claim 11, wherein during system shutdown, the microcontroller is further configured to:

configure the non-volatile memory to allow access to the non-volatile memory only by the microcontroller before power-off;

queue one or more writes or other exchanges with the non-volatile memory; and

complete the one or more writes or other exchanges with the non-volatile memory before final power-off.

166. The system of claim 11, further comprising:

a device interface coupled to the non-volatile memory, wherein the device interface controls access to the non-volatile memory, wherein during system shutdown, the microcontroller is further configured to:

redirect the device interface to allow access to the non-volatile memory only by the microcontroller before power-off;

queue one or more writes or other exchanges with the non-volatile memory; and

complete the one or more writes or other exchanges with the non-volatile memory before final power-off.

167. The system of claim 11, further comprising:

one or more additional processors, each coupled to the microcontroller and the non-volatile memory, and wherein the microcontroller has control over the power or execution states of the processor and the one or more additional processors for sharing the non-volatile memory between the plurality of processors.

168. The system of claim 11, further comprising:

one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, wherein each additional microcontroller is configured to share the non-volatile memory with the processor.

169. The system of claim 11, further comprising:

one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, wherein each additional microcontroller is configured to share the non-volatile memory, and wherein each microcontroller is configured to suspend or prevent access to the non-volatile memory by other microcontrollers and the processor while it accesses the non-volatile memory.

170. The system of claim 11, further comprising:

a system interface, coupled to the processor, wherein the microcontroller is situated between the system interface and the non-volatile memory, and wherein the microcontroller is configured to intercept communications between the system interface and the non-volatile memory; and

a device interface, included in or associated with the non-volatile memory, wherein the device interface is configured to direct or restrict access to the non-volatile memory by the processor or the microcontroller.

171. The system of claim 170, wherein the microcontroller is configured to set one or more control signals to configure or redirect the device interface to grant exclusive access to the non-volatile memory to the microcontroller.

172. The system of claim 11, wherein the non-volatile memory comprises flash memory.

173. The system of claim 11, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller, the microcontroller is configured to:

fetch initialization code or data for the system.

174. The system of claim 173, wherein to fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller, the microcontroller is further configured to:

fetch program instructions or data related to pre-boot security operations; and  
perform the pre-boot security operations prior to boot-up of the system.

175. The system of claim 11, wherein the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus.

176. The system of claim 175, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable to switchably intercept or pass communications between the processor and the non-volatile memory.

177. The system of claim 176, wherein the host memory bus and the non-volatile memory bus are SPI (serial peripheral interface) buses.

178. The system of claim 176, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor and the non-volatile memory.

179. The system of claim 178, wherein the switch comprises a Q switch or a conceptual Q switch.

180. The system of claim 178, wherein to switchably intercept or pass communications between the processor and the non-volatile memory, the microcontroller is configured to place the switch into a high-impedance state.

181. The system of claim 176, wherein the host memory bus and the non-volatile memory bus each comprises:

- a data line;
- a clock line; and
- a CSb (Chip Select bar) line.

182. The system of claim 176, wherein the host memory bus and the non-volatile memory bus each comprises a plurality of lines.

183. The system of claim 182, wherein each line of the host memory bus and the non-volatile memory bus is independently switchable.

184. The system of claim 182, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller for communicating signals between the system interface and the microcontroller;

wherein the lines of the host memory bus and the non-volatile memory bus are switchable together to suspend or inhibit communications between the system interface and the non-volatile memory.

185. The system of claim 182, wherein the lines of the host memory bus and the non-volatile memory bus are coupled to respective buffers for communicating data or signals between the non-volatile memory bus and the microcontroller.

186. The system of claim 11, further comprising:

a system interface, coupled to the processor, and further coupled to the microcontroller via a system management bus configured for communicating management signals between the system interface and the microcontroller;

wherein the microcontroller comprises a system reset line for communicating the system reset signal or other signal to or from the system interface.

187. The method of claim 17, wherein the microcontroller is included within a system-on-chip (SoC).

188. The method of claim 17, wherein the microcontroller comprises a keyboard controller for managing or controlling communications between a keyboard and the processor.

189. The method of claim 188, wherein the non-volatile memory stores keyboard controller (KBC) code.

190. The method of claim 17, wherein the program instructions or data are usable by the microcontroller for startup, general, or reset functionality.

191. The method of claim 17, wherein the program instructions or data are usable by the microcontroller for initializing the microcontroller.

192. The method of claim 17, further comprising:

during system shutdown:

configuring the non-volatile memory to allow access to the non-volatile memory only by the microcontroller before power-off;

queuing one or more writes or other exchanges with the non-volatile memory; and  
completing the one or more writes or other exchanges with the non-volatile memory before final power-off.

193. The method of claim 17, further comprising:

during system shutdown:

redirecting a device interface to allow access to the non-volatile memory only by the microcontroller before power-off, wherein the device interface is coupled to the non-volatile memory and controls access to the non-volatile memory;

queuing one or more writes or other exchanges with the non-volatile memory; and



completing the one or more writes or other exchanges with the non-volatile memory before final power-off.

194. The method of claim 17, wherein the system further includes one or more additional processors, each coupled to the microcontroller and the non-volatile memory, and wherein the method further includes controlling the power or execution states of the processor and the one or more additional processors for sharing the non-volatile memory between the plurality of processors.

195. The method of claim 17, wherein the system further includes one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, and wherein the method further includes each additional microcontroller sharing the non-volatile memory with the processor.

196. The method of claim 17, wherein the system further includes one or more additional microcontrollers, each coupled to the processor and the non-volatile memory, and wherein the method further includes:

each additional microcontroller sharing the non-volatile memory; and  
each microcontroller suspending or preventing access to the non-volatile memory by other microcontrollers and the processor while it accesses the non-volatile memory.

197. The method of claim 17, further comprising:

intercepting communications between a system interface and the non-volatile memory, wherein the system interface is coupled to the processor, and wherein the microcontroller is situated between the system interface and the non-volatile memory; and  
a device interface directing or restricting access to the non-volatile memory by the processor or the microcontroller, wherein the device interface is included in or associated with the non-volatile memory.

198. The method of claim 197, wherein the microcontroller is configured to set one or more control signals to configure or redirect the device interface to grant exclusive access to the non-volatile memory to the microcontroller.

199. The method of claim 17, wherein the non-volatile memory comprises flash memory.

200. The method of claim 17, wherein said fetching program instructions or data from the non-volatile memory further comprises fetching initialization code or data for the system.

201. The method of claim 200, further comprising:  
fetching program instructions or data related to pre-boot security operations; and  
performing the pre-boot security operations prior to boot-up of the system.

202. The method of claim 17, wherein fetching the program instructions or data from the non-volatile memory occurs via a non-volatile memory bus that couples the microcontroller and the non-volatile memory.

203. The method of claim 202, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the method further comprises the microcontroller switchably intercepting or passing communications between the processor and the non-volatile memory.

204. The method of claim 203, wherein the host memory bus and the non-volatile memory bus are SPI (serial peripheral interface) buses.

205. The method of claim 203, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, and wherein the microcontroller switchably intercepting or passing communications comprises the microcontroller controlling the switch to switchably intercept or pass communications between the processor and the non-volatile memory.

206. The method of claim 205, wherein the switch comprises a Q switch or a conceptual Q switch.

207. The method of claim 205, wherein switchably intercepting communications between the processor and the non-volatile memory comprises placing the switch into a high-impedance state.

208. The method of claim 203, wherein the host memory bus and the non-volatile memory bus each comprises:

- a data line;
- a clock line; and
- a CSb (Chip Select bar) line.

209. The method of claim 203, wherein the host memory bus and the non-volatile memory bus each comprises a plurality of lines.

210. The method of claim 209, wherein each line of the host memory bus and the non-volatile memory bus is independently switchable.

211. The method of claim 209, further comprising:

switching the lines of the host memory bus and the non-volatile memory bus together to suspend or inhibit communications between a system interface and the non-volatile memory, wherein the system interface is coupled to the processor, and further coupled to the microcontroller for communicating signals between the system interface and the microcontroller.

212. The method of claim 209, wherein the lines of the host memory bus and the non-volatile memory bus are coupled to respective buffers for communicating data or signals between the non-volatile memory bus and the microcontroller.

213. The method of claim 17, further comprising:

communicating management signals between a system interface and the microcontroller, wherein the system interface is coupled to the processor and is further coupled to the microcontroller via a system management bus, and wherein the microcontroller comprises a system reset line for communicating the system reset signal or other signal to or from the system interface.

## **REMARKS**

Claims 1-23 were originally issued in U.S. Patent 7,930,576 (the “576 patent”). Claims 24-213 have been added, and are fully supported by the original specification:

Claims 24-31 and 89-96 are fully supported by at least col.5, line 58 – col. 6, line 7 and col. 15, lines 50-67;

Claims 32-34 and 97-99 are fully supported by at least original claim 3;

Claims 35-37 and 100-102 are fully supported by at least original claim 11;

Claims 38 and 103 are fully supported by at least col. 10, lines 51-62;

Claims 39-43, 104-108, 160-164, and 187-191 are fully supported by at least col. 7, lines 32-41, col. 13, lines 42-48, and col. 8, lines 9-23;

Claims 44-48, 109-113, 165-169, and 192-196 are fully supported by at least col. 10, line 63 – col. 11, line 14 and col. 6, lines 56-62;

Claims 49-50, 114-115, 170-171, and 197-198 are fully supported by at least col. 8, line 24 – col. 9, line 46;

Claims 51-63, 116-128, 172, and 199 are fully supported by at least col. 9, line 47 – col. 10, line 3;

Claims 64-65, 129-130, 173-174, and 200-201 are fully supported by at least col. 14, line 60 – col. 15, line 6;

Claims 66-73, 131-138, 175-180, and 202-207 are fully supported by at least col. 4, line 60 – col. 5, line 20, col. 4, lines 4-46, and col. 14, lines 1-26;

Claims 74, 139, 181, and 208 are fully supported by at least col. 13, lines 49-67;

Claims 75-79, 140-144, 182-185, and 209-212 are fully supported by at least col. 12, lines 8-20 and col. 14, lines 1-26;

Claims 80-82 and 145-147 are fully supported by at least col. 14, lines 43-49;

Claims 83-84 and 148-149 are fully supported by at least col. 9, lines 4-21;

Claims 85-88, 150-153, 186, and 213 are fully supported by at least col. 10, lines 21-40;

Claims 154-159 are fully supported by at least original claim 1.

Therefore, claims 1-213 are now pending in this reexamination.

## I. Introduction

The instant Office Action rejects originally issued claims 1-23 of the '576 patent as allegedly being either anticipated by or obvious in view of U.S. Patent 6,161,162 to DeRoo, et al. ("DeRoo"). In the following discussion, Patent Owner demonstrates that these rejections are in error, and that each of claims 1-23 as well as each of the newly added claims listed above is patentable over DeRoo.

Below, Patent Owner first briefly summarizes the disclosures of both the '576 patent and DeRoo. Next, Patent Owner demonstrates that the Office Action's position concerning the correspondence of the recited "microcontroller" to both DeRoo's SCP 706 and HUI 700 taken together does not comport with the meaning of the term "microcontroller" as it is used in the '576 patent, even under the broadest reasonable construction. In addressing the anticipation rejections, Patent Owner further demonstrates DeRoo fails to disclose numerous features of independent claims 1 or 6. First, DeRoo does not disclose "a first state wherein the microcontroller is assured safe access to the non-volatile memory." DeRoo also fails to disclose that a microcontroller is configured to "hold the system in the first state." DeRoo additionally fails to disclose that a microcontroller is configured to "release the system from the first state." DeRoo further fails to disclose that a microcontroller is configured to "switch access to the non-volatile memory from the processor to the microcontroller" "in response to a change in system state to a first state wherein the microcontroller is assured safe access."

In addressing the obviousness rejections, Patent Owner demonstrates that the modification of DeRoo proposed in the Office Action cannot sustain a *prima facie* case of obviousness. Specifically, the Office Action proposes that although DeRoo does not disclose that a system control processor (SCP) "hold[s] a system reset signal in a reset state, thereby prohibiting [a] processor from accessing [a] non-volatile memory," as recited in independent claims 11 and 17, the Office Action states that it would have been obvious to modify DeRoo's SCP to reset DeRoo's CPU. However, as Patent Owner shows below, such a modification would change the principle of operation of DeRoo, which is predicated upon a system in which the CPU has unconditionally higher priority than the SCP (which would no longer be the case if the

SCP could reset the CPU as proposed). Moreover, such a modification would render DeRoo unsatisfactory for its intended purpose of use within a personal computer, because it would increase the susceptibility of DeRoo's system to undesirable behavior such as system hangs and spontaneous rebooting. Individually, either of these consequences of the proposed modification would be sufficient to preclude such a modification as a basis for an obviousness rejection; the fact that both consequences arise leaves little doubt that the proposed modification is impermissible.

Patent Owner respectfully submits that the present rejections should be withdrawn, and all claims confirmed as patentable, for at least the reasons given below.

## II. Overview of '576 Patent

The '576 patent is directed to techniques for sharing devices, such as a non-volatile memory, among multiple processors in a system. *See* '576 patent, col. 1, ll.16-20. Specifically, it is common for hardware systems to include a central processing unit (CPU) as well as one or more microcontrollers for performing auxiliary system functions. *See id.*, ll.21-25. Typically, each processor or microcontroller in a system relies on its own set of firmware instructions for controlling operations such as initialization, power management, or other functions. *See id.*, ll.25-30. Such firmware instructions are often stored in non-volatile memory such as flash memory, so that they may be updated during the manufacturing process or in the field. *See id.*, ll.51-53. However, providing each processor and microcontroller with its own discrete non-volatile memory in which to store its firmware is costly, as is embedding the non-volatile memory within the processor or microcontroller itself. *See id.*, ll.37-39, 53-60.

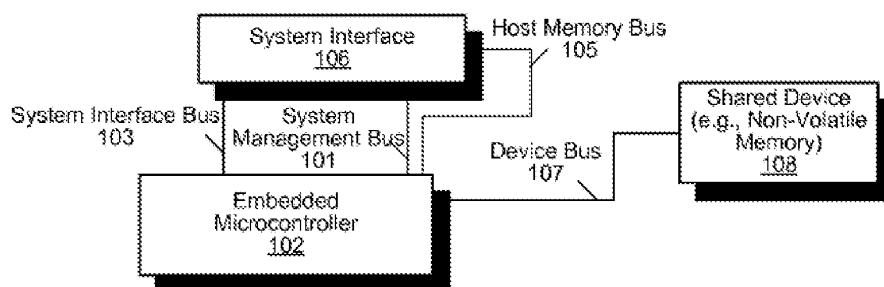
An alternative to providing each processor and microcontroller with its own discrete or embedded non-volatile storage is to design the system such that the processor(s) and microcontroller(s) arbitrate for access to a shared memory device. *See* '576 patent, col. 1, ll.33-34. However, arbitration adds design and hardware complexity (since arbitration must be controlled by an arbiter), and arbitration may degrade system performance and user experience (e.g., because of delays associated with arbitration). *See id.*, ll.35-37.

In order to avoid the costs and complexities of these solutions, the '576 patent proposes a system having at least a processor and a microcontroller, in which a microcontroller is provided with a system state in which it is assured safe access to a shared memory device, or where the microcontroller can prevent the system from entering a state where the processor has access to the shared memory device. *See* '576 patent, col. 2, ll.22-25. That is, the system may provide a state (also referred to as a "first" state) in which "the microcontroller is guaranteed the ability to access the shared device safely, e.g., without interruption or interference by the processor." *See id.* ll.27-29. During this state, the microcontroller may access the shared memory device to retrieve or exchange data, such as firmware. *See id.*, col. 3, ll.1-3. Once its access to the shared memory device is complete, the microcontroller may "change or allow change of the system



state, thereby permitting the processor in the system to access the shared device.” *See id.*, ll.22-25.

One example of a system configuration using the techniques of the ’576 patent is shown in Figure 1B, reproduced below.



In the illustrated embodiment, embedded microcontroller 102 is coupled to system interface 106, which may simply be a bus or other interface suitable for communicating between microcontroller 102 and a CPU, or may include other structure or functionality. *See* ’576 patent, col. 7, ll.59-66. Microcontroller 102 is further coupled to shared device 108, which may be a non-volatile memory that stores firmware for both microcontroller 102 and a CPU located behind system interface 106. *See id.*, col. 8, ll.1-15.

In some embodiments, the “first state” may correspond to a reset state, such as a power-on reset. *See* ’576 patent, col. 8, ll.65-67. However, the “first state” may also correspond to other states, such as a system reset state or a sleep state. *See id.* That is, the microcontroller may take advantage of any of a variety of states in which it is assured safe access to the shared memory in order to safely use the shared memory. *See id.*, col. 3, ll.38-45.

Thus, using the techniques of the ’576 patent,

the microcontroller may preempt or suspend access to the device by the processor, e.g., the main CPU, of the system, or otherwise configure safe access to the device by the microcontroller; retrieve or otherwise exchange data or program instructions with the device; then return access to the device to the processor, or otherwise relinquish exclusive or safe access to the device by the microcontroller.

*See* '576 patent, col. 3, ll.30-37. As a result, “the microcontroller may gracefully share access to the device with the processor of the system.” *See id.*, ll.58-59.

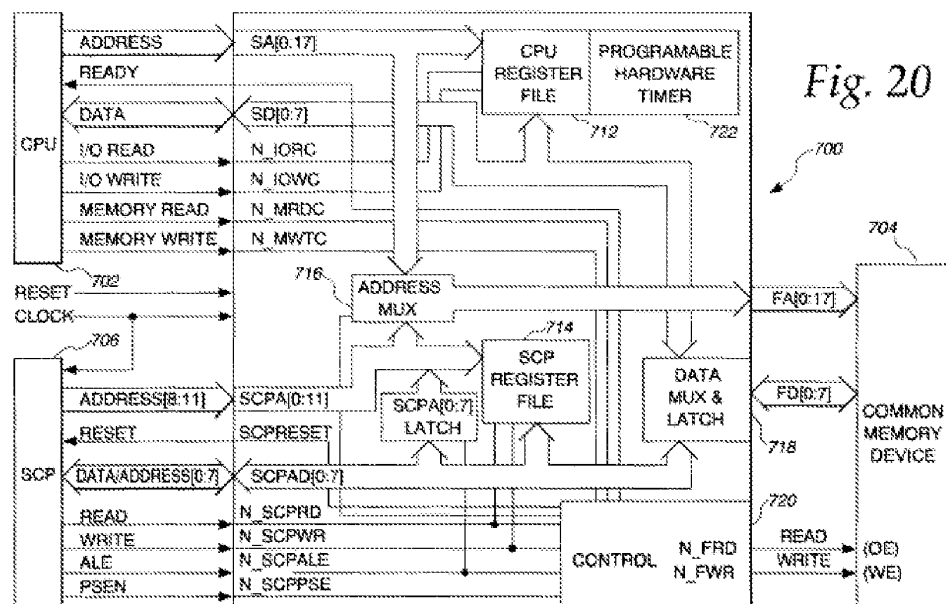
As discussed in greater detail in Section IV.A.2.a, *infra*, one of ordinary skill in the art would understand that, consistent with the specification of the '576 patent, and consistent with the plain meaning of the phrase “assured safe access,” for a microcontroller to be “assured safe access” to a shared memory, under normal operating conditions, the microcontroller must be guaranteed access to the shared memory in a manner that is free from interruption or interference from a processor. Specifically, one of ordinary skill would understand that during a state in which a microcontroller is “assured safe access” to a shared memory, at a minimum, the microcontroller’s access cannot be arbitrarily preempted by the processor in such a way that the microcontroller could be adversely affected. *See* Declaration of Thomas M. Conte Under 37 C.F.R. § 1.132 (“132 Declaration”) at ¶ 12. Thus, one of ordinary skill would recognize the system of the '576 patent as a system in which the microcontroller can freely preempt the processor of the system, but not vice versa, and thus as a system in which the microcontroller has higher access priority than the processor with respect to the shared memory. *See* 132 Declaration at ¶ 13.

### III. Overview of the DeRoo Reference

U.S. Patent 6,161,162 to DeRoo et al. (“DeRoo”), entitled “Multiprocessor System for Enabling Shared Access to a Memory,” relates to “providing multiplexed address and data paths from multiple CPUs to a single storage device,” with the object being “to allow the use of a single memory device to serve as a code storage medium for multiple CPUs in a digital computer system.” *See* DeRoo, col. 2, ll. 13-15, 1-3.

DeRoo purports to disclose “[t]hree embodiments of the invention” in which “[t]he first embodiment relates to a System Control Processor Interface (SCPI)” as shown in Figs. 2-9, “[t]he second embodiment relates to a Mouse Keyboard Interface (MKI)” as shown in Figs. 10-19, and “[t]he third embodiment relates to a Human User-Input Interface (HUI),” as shown in Figs. 20-35. *See id.*, col. 3, l.64 – col. 4, l.2. However, only the HUI embodiment of DeRoo is actually addressed to the problem of sharing access to a memory device: “The HUI solves this problem by providing an interface to enable both the SCP and the CPU to share a common memory device and, additionally, incorporates the SCPI and MKI interfaces.” *See id.*, col. 36, ll.30-33. The Office Action relies only on the HUI embodiment of DeRoo, specifically citing to columns 36-37 and 81-84 relating to the HUI embodiment.

DeRoo describes the HUI embodiment as “an application specific integrated circuit (ASIC) which integrates the SCPI and MKI interfaces . . . into a single device along with an interface for a common memory device, which may be reprogrammable, for a central processing unit (CPU) and a system control processor (SCP).” *See* DeRoo, col. 36, ll.1-6. As DeRoo states, “[t]he common memory device enables the basic input/output system (BIOS) for the CPU, as well as the firmware for the SCP to be stored in the same memory device.” *See id.*, ll.6-9. DeRoo illustrates a block diagram of the HUI embodiment in Fig. 20, reproduced below, which specifically shows HUI 700, CPU 702, SCP 706, and common memory device 704:



DeRoo explains that SCP 706 is a microcontroller: “the firmware for the SCP . . . may be onboard the microcontroller for the SCP depending on the particular microcontroller selected for the SCP”; “Intel type 80C51 microcontrollers are known to be used for the SCP”; “[O]ther microcontrollers, such as an Intel type 80C31 are known to be used [for the SCP].” *See* DeRoo, col. 36, ll.17-27. DeRoo does not appear to explicitly identify what type of processor is used as CPU 702 in the HUI embodiment. However, DeRoo states with respect to the distinct SCPI embodiment that “an Intel 80286, 80386 or an 80486 is used as a CPU.” *See id.*, col. 4, ll.26-27.

DeRoo provides three distinct modes of operation with respect to common memory device 704, each of which will be explained below in greater detail: a CPU-exclusive mode in which CPU 702 has exclusive access to memory device 704, an SCP-exclusive mode in which SCP 706 has exclusive access to memory device 704, and a shared mode in which CPU 702 and SCP 706 share access to memory device 704. *See generally* DeRoo, cols. 81-84.

As noted by DeRoo, “[e]xclusive access by the CPU 702 is required to write to the common memory device 704.” *See* DeRoo, col.82, ll.21-22. In order to enter CPU-exclusive

mode, DeRoo repeatedly states that CPU 702 “can gain exclusive control . . . by placing the SCP into a reset state,” noting that “[d]uring such conditions, the SCP 706 is prevented from writing to the common memory device 704 since the SCP is held in a reset state during such conditions.” *See id.* at col. 82, ll.1-3, 29-31; *see also* col. 81, ll.46-49 (noting that “SCP 706 [is prevented] from gaining access to the common memory device 704 during periods when the CPU 702 has exclusive control since, during such conditions, the SCP 706 is held in reset.”).

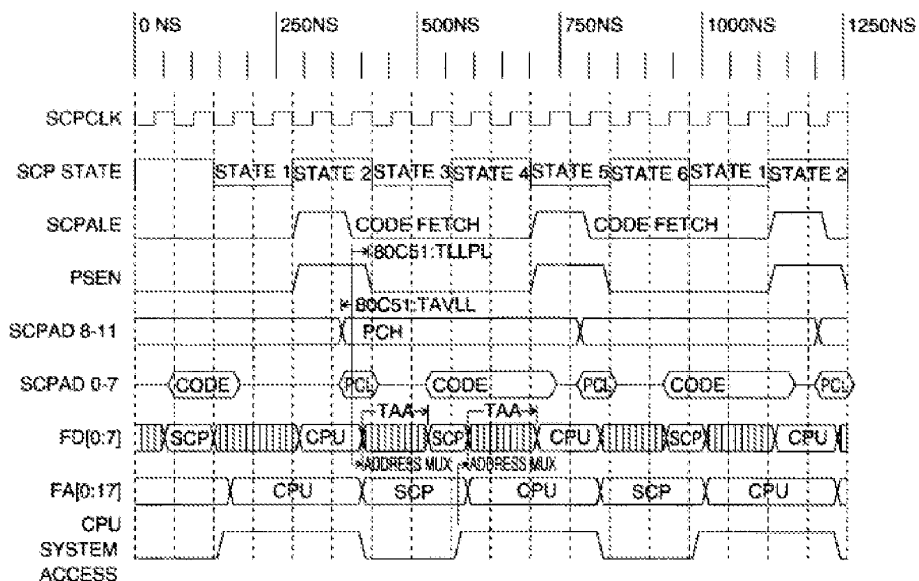
Moreover, DeRoo unequivocally states that CPU 702 may unconditionally gain access to CPU-exclusive mode even if the SCP currently has exclusive access: “During conditions when the SCP has exclusive access of the common memory device 704, the CPU can gain control by forcing the SCP 706 into reset.” *See* DeRoo, col. 81, ll.64-66 (emphasis added). DeRoo’s description of the control register through which CPU 702 asserts its demand for exclusive access bears this out: in describing the CPUEXCL flag of the FLASH\_CTRL register, DeRoo states that this flag indicates that “SCP has requested exclusive access to flash memory for a write and CPU is disallowed from reading and writing the FLASH - this bit can be cleared immediately if a ‘1’ is written to SCPreset.” *See id.*, col. 49, Table XXXI (emphasis added). During conditions when SCP 706 does not have exclusive access, DeRoo states that the mechanism by which CPU 702 gains access to CPU-exclusive mode is the same as when SCP 706 does have exclusive access: “by placing the SCP 706 into a reset state by setting bit 0 of the flash interface control register FLASH\_CTRL.” *See id.*, col. 82, ll.2-4.

As with CPU 702, DeRoo notes that “SCP 706 can only write to the common memory device 704 when it has exclusive access.” *See* DeRoo, col. 82, ll.33-34. However, there is a clear asymmetry between SCP 706 and CPU 702 with respect to their abilities to gain exclusive access: unlike CPU 702, which is able to force SCP 706 into reset, SCP 706 cannot force CPU 702 into reset. Instead, “[i]n order for the SCP 706 to gain exclusive control, the CPU 702 may be forced into a wait state by the HUI 700 by pulling the input/output channel ready signal IOCHRDY inactive.” *See id.*, col. 81, ll.55-58. The IOCHRDY signal is a standard feature of the Industry Standard Architecture/Extended Industry Standard Architecture (ISA/EISA) bus interface (e.g., as implemented by the original IBM PC) that indicates to a bus master—in DeRoo’s case, CPU 702—that additional cycle time is required and that the bus master should

wait before accessing the ISA/EISA bus. *See* 132 Declaration at ¶ 18. (The IOCHRDY signal behaves nothing like a reset signal; in fact, if it did, DeRoo's system would fail to operate properly, as demonstrated in greater detail below.) As noted above, even when SCP 706 gains exclusive control, this control does not amount to "assured safe access," because CPU 706 can arbitrarily preempt SCP 706 and gain control of the shared memory whenever it desires to do so.

DeRoo provides a third, shared mode in which both CPU 702 and SCP 706 are given access to common memory device 704 according to a "time based interleaving method." *See* DeRoo, col. 81, l.51. During this mode, "common memory device 704 is configured as a read only memory," consistent with the requirement that writing to memory device 704 requires exclusive access. *See id.*, col. 82, ll.43-44. The general timing of the interleaved access that occurs during DeRoo's shared mode is illustrated in Fig. 24, reproduced below:

Fig. 24



DeRoo indicates that in shared mode, "[t]he SCP 706 has control for about 2.5 to 3.5 SCP clock periods, after which time the CPU 702 is given control." *See* DeRoo, col. 83, ll.48-50.

Specifically, DeRoo notes that in shared mode, once SCP 706 has been granted access, the amount of time SCP 706 has access to common memory device 704 is fixed and invariant: “Access is granted to the SCP 706 as soon as the SCP address latch enable signal SCPALE occurs (start of an SCP cycle) and switched to the CPU 702 after two rising and one falling edges of the SCP clock SCPCLK are detected.” *See id.*, ll.53-57 (emphasis added).

That is, DeRoo’s shared mode is implemented as a conventional time division multiplexed operation in which, so long as the shared mode is active, SCP 706 and CPU 702 share access to common memory device 704 according to a predictable schedule that is controlled not by either of these devices, but is instead independently controlled by HUI 700. *See* 132 Declaration at ¶ 20. This is further evidenced by DeRoo’s Verilog source code appendix that describes the functionality of HUI 700, in which the signal “flashSCP” (which determines whether the SCP has access to the common memory device 704) is set by the falling edge of signal “SCPALE” and is cleared by the rising edge of signal “clr\_flashSCP,” which is itself merely a version of the “flashSCP” signal that has been passed through a fixed delay chain. *See* DeRoo, cols. 129, 131.

It is clear from DeRoo’s description that regardless of whether the system is operating in shared mode or SCP-exclusive mode, CPU 702 may demand and receive exclusive access to common memory device 704 at its prerogative. As noted above, CPU 702 may gain exclusive control of common memory device 704 even when SCP 706 has exclusive access by “forcing the SCP 706 into reset.” *See* DeRoo, col. 81, ll. 66. Even when SCP 706 does not have exclusive access, CPU 702 “can gain exclusive control . . . by placing the SCP 706 into a reset state.” *See id.*, col. 82, ll. 1-3. DeRoo only discloses three possible modes of operation, one of which is a CPU-exclusive mode. Because CPU 702 would not need to “gain exclusive control” while in CPU-exclusive mode, DeRoo’s latter reference to CPU 702 “gain[ing] exclusive control” necessarily refers to shared mode. *See* 132 Declaration at ¶ 21.

In summary, DeRoo does in fact provide a system through which CPU 702 and SCP 706 may share access to common memory device 704. However, DeRoo’s system is organized around the basic principle that CPU 702 can preempt SCP 706’s access to memory device 704 at

any time by forcing SCP 706 into reset, regardless of whether SCP 706 has exclusive access to memory device 704 or whether CPU 702 and SCP 706 are sharing access to memory device 704 in a time-multiplexed fashion. *See* 132 Declaration at ¶ 22. That is, in DeRoo, CPU 702 is effectively a master device that can reset SCP 706 at any time, whereas SCP 706 is a slave device that can request exclusive access to memory device 704, but will only hold such exclusive access—or even shared access—for so long as CPU 702 allows. *See id.* Consequently, DeRoo describes a system in which a processor (CPU 706) unconditionally holds higher access priority than a microcontroller (SCP 702) with respect to a shared memory device (common memory device 704). *See id.*



#### IV. Response to Rejections Based on DeRoo

Claims 1, 2, 4-7, 9 and 10 are rejected under 35 U.S.C. § 102(b) based on DeRoo.<sup>1</sup> Claims 3, 8, and 11-23 are rejected under 35 U.S.C. § 103(a) as being unpatentable over the single reference of DeRoo. Patent Owner traverses these rejections for at least the reasons set forth below. Rejections under 35 U.S.C. § 102(b) are addressed first, followed by rejections under 35 U.S.C. § 103(a).

Rejections under 35 U.S.C. § 102(b):

##### A. CLAIM 1:

As an initial matter, the Office Action's position concerning the correspondence of the recited "microcontroller" to both DeRoo's SCP 706 and HUI 700 taken together does not comport with the meaning of the term "microcontroller" as it is used in the '576 patent, even under the broadest reasonable construction. Moreover, DeRoo fails to disclose numerous features of claim 1. First, DeRoo does not disclose "a first state wherein the microcontroller is assured safe access to the non-volatile memory." DeRoo also fails to disclose that a microcontroller is configured to "hold the system in the first state." DeRoo additionally fails to disclose that a microcontroller is configured to "release the system from the first state." DeRoo further fails to disclose that a microcontroller is configured to "switch access to the non-volatile memory from the processor to the microcontroller" "in response to a change in system state to a first state wherein the microcontroller is assured safe access."

Patent Owner therefore traverses the rejection of claim 1 as set forth in detail below.

---

<sup>1</sup> In the Office Action, the proposed rejection under 35 U.S.C. § 102(b) was changed *sua sponte* to a rejection under 35 U.S.C. § 102(e). See Office Action at 4. Specifically, the Office Action states that "DeRoo was not published until December 12, 2000, which was after the December 18, 2007 filing date of the '576 patent." *Id.* Patent Owner submits that in fact, these dates evidence that DeRoo was published before the filing date of the '576 patent, and thus a rejection under § 102(e) would not be appropriate here. For the purposes of this response, Patent Owner will treat the DeRoo reference as a § 102(b) reference.

1. The combination of DeRoo's SCP 706 and HUI 700 does not correspond to the recited "microcontroller."

In rejecting claim 1, the Office Action alleges that DeRoo's common memory device 704 corresponds to the recited "non-volatile memory," that DeRoo's CPU 702 corresponds to the recited "processor," and that DeRoo's SCP 706 and HUI 700 together correspond to the recited "microcontroller." See Office Action at 4. As an initial matter, Patent Owner submits that the Office Action's position concerning the recited "microcontroller" is inconsistent with the meaning of that term as it is employed in the '576 patent—i.e., the plain meaning of that term as it would be understood by one of ordinary skill in the art. Moreover, Patent Owner submits that the Office Action's position is inconsistent with DeRoo's own usage of the term.

Claim terms are to be given their broadest reasonable interpretation (BRI) consistent with the specification. See *Phillips v. AWH Corp.*, 415 F.3d 1303, 1316-17 (Fed. Cir. 2005) (*en banc*) ("The Patent and Trademark Office ('PTO') determines the scope of claims in patent applications not solely on the basis of the claim language, but upon giving claims their broadest reasonable construction 'in light of the specification as it would be interpreted by one of ordinary skill in the art.'"). See also M.P.E.P. 2111 ("During patent examination, the pending claims must be 'given their broadest reasonable interpretation consistent with the specification.'").

Under BRI, the term "microcontroller" may encompass many types of processing devices. However, the breadth of this term is not unlimited. One of ordinary skill in the art would readily recognize that, consistent with its usage in the '576 patent, the term "microcontroller" refers to a single integrated circuit device—i.e., to a "chip" fabricated as part of a separately packaged device. See 132 Declaration at ¶ 24. Although dictionary definitions of "microcontroller" vary, numerous definitions confirm this "single-chip" aspect of the term. See, e.g., Webopedia, available at <http://www.webopedia.com/TERM/M/microcontroller.html> (referring to a "highly integrated chip"); Microsoft Computer Dictionary, Fifth Edition, at 337 (referring to a "single chip computer"); PC Magazine Encyclopedia, available at [http://www.pcmag.com/encyclopedia\\_term/0,2542,t=microcontroller&i=46924,00.asp](http://www.pcmag.com/encyclopedia_term/0,2542,t=microcontroller&i=46924,00.asp) (referring to "a single chip"). The meaning of "microcontroller" as referring to a single integrated circuit device is consistent with usage of the term "microcontroller" in the '576 patent. For example, in

describing the difficulties of incorporating embedded flash into a microcontroller, the '576 patent notes the complexities of integrating flash memory “on the same die” as the “microcontroller chip.” *See* '576 patent, col. 1, ll.54-56; *see also* 132 Declaration at ¶ 25.

DeRoo also understands “microcontroller” to refer to a separate integrated circuit device that is distinct from, and does not include, DeRoo’s HUI device. Specifically, DeRoo identifies HUI 700 as “an application specific integrated circuit (ASIC) which integrates the SCPI and MKI interfaces discussed above into a single device along with an interface for a common memory device, which may be reprogrammable, for a central processing unit (CPU) and a system control processor (SCP).” *See* DeRoo, col. 36, ll.3-7 (emphasis added); *see also* 132 Declaration at ¶ 27. DeRoo also repeatedly refers to the SCP itself as both a “microcontroller” and as a discrete integrated circuit device. Specifically, DeRoo states that

the firmware for the SCP is normally stored in a separate memory device, which may be onboard the microcontroller for the SCP depending on the particular microcontroller selected for the SCP or in a separate memory device. For example, Intel type 80C51 microcontrollers are known to be used for the SCP. Such microcontrollers have on-board internal ROM for storing the firmware for the SCP and are relatively more expensive and considerably more difficult to field upgrade. Alternatively, other microcontrollers, such as an Intel type 80C31 are known to be used which do not contain on-board ROM.

*See* DeRoo, col. 36, ll.16-27 (emphasis added).

That is, one of ordinary skill would understand DeRoo to employ the term “microcontroller” to refer to SCP 706, and would further understand DeRoo to employ the term “microcontroller” in its ordinary sense of referring to a discrete integrated circuit device—a conclusion supported by DeRoo’s specific references to examples of 80C51 and 80C31 microcontrollers, which are themselves single-chip microcontrollers. *See* 132 Declaration at ¶ 26. Moreover, one of ordinary skill would understand DeRoo’s HUI 700 to be a discrete ASIC corresponding to a device that is distinct from SCP 706, and thus distinct from a “microcontroller.” *See* 132 Declaration at ¶ 27.

Accordingly, given that the ordinary meaning of “microcontroller,” even under BRI, refers to a single integrated circuit device, and that DeRoo plainly describes SCP 706 and HUI

700 as distinct integrated circuit devices, the Office Action's allegation that SCP 706 and HUI 700 together correspond to the recited "microcontroller" is inconsistent with the ordinary meaning of that term as it is used both in the '576 patent and in DeRoo. That is, it is inappropriate to ascribe the features of one discrete device (HUI 700) to another (SCP 706) and to then call the result a "microcontroller," especially where DeRoo clearly considers SCP 706 itself to be a "microcontroller." *See* 132 Declaration at ¶ 28.

2. DeRoo fails to disclose "a first state wherein the microcontroller is assured safe access to the non-volatile memory."

The Office Action alleges that the recited "first state" is disclosed by DeRoo's holding the CPU SYSTEM ACCESS signal in a low level, as shown in DeRoo's FIG. 24. *See* Office Action at 5. Patent Owner respectfully disagrees and submits that for at least the following reasons, there is no operational state of DeRoo's system for sharing a common memory device in which a microcontroller is assured safe access to the memory device.

*a. For a microcontroller to be "assured safe access" to a memory, under a broadest reasonable interpretation of that term, the microcontroller's access to the memory cannot be arbitrarily preempted by a processor.*

Patent Owner submits that, under the BRI standard for claim construction discussed in the previous section, the meaning of the phrase "assured safe access" is governed by the ordinary meaning of its terms consistent with their usage in the specification. Specifically, the ordinary meaning of "assured" that is consistent with its usage in the '576 patent is "characterized by certainty or security: guaranteed." *See* Merriam-Webster Online Dictionary, <http://www.merriam-webster.com/dictionary/assured>. In describing the "first state," the '576 patent provides explicit evidence that as used in this context, "assured" means "guaranteed" under normal operating conditions: "In other words, the system may be changed to a state (a 'first' state) in which the microcontroller is guaranteed the ability to access the shared device safely." *See* '576 patent, col. 2, ll.27-29 (emphasis added).

Moreover, the ordinary meaning of “safe” that is consistent with its usage in the ’576 patent is “free from harm or risk; secure from threat of danger, harm, or loss.” *See* Merriam-Webster Online Dictionary, <http://www.merriam-webster.com/dictionary/safe>. The ’576 patent provides further guidance as to the meaning of “safe” in the context of access by a microcontroller: that is, “safe access” occurs “without interruption or interference by the processor.” *See* ’576 patent at col. 2, l.30.

The phrase “assured safe access” is specifically used in the claims to characterize the access of a microcontroller to a non-volatile memory in the context of a system that also includes a processor. When construed consistently with the plain meaning of its terms as set forth above, for a microcontroller to be “assured safe access” to a non-volatile memory in a system where a processor can also access the non-volatile memory, under normal operating conditions, the microcontroller must be guaranteed access to the non-volatile memory in a manner that is free from interruption or interference by the processor.<sup>2</sup>

One of ordinary skill in the art would understand that for a microcontroller to be “assured safe access” to a non-volatile memory relative to a processor in this context, it is necessary to provide a mechanism to ensure that the processor cannot arbitrarily preempt the microcontroller’s access to the non-volatile memory. *See* 132 Declaration at XX. That is, for a microcontroller to be “assured safe access,” its access must be exclusive to the microcontroller and not preemptible by the processor in a manner that could adversely affect the microcontroller. This follows because if the microcontroller’s access to the non-volatile memory were not exclusive and could be preempted by the processor without warning, the processor could arbitrarily interfere with the microcontroller’s access in a way that affects the microcontroller (e.g., by preventing the microcontroller from completing its access or changing the contents of the memory on which the microcontroller depends) and the microcontroller’s access therefore could not be “assured” to be “safe.” *See* 132 Declaration at ¶¶ 12-13, 29.

*b. In DeRoo's system, CPU 702 can arbitrarily preempt SCP 706 at any time, which is precisely the opposite of what is needed for a microcontroller to be "assured safe access" to a memory.*

As noted above, the Office Action alleges that the recited "first state" in which the microcontroller is "assured safe access" to a memory is the state in which the CPU SYSTEM ACCESS signal is at a low level, as shown in DeRoo's Figure 24. *See* Office Action at 5. DeRoo describes Figure 24 as "a timing diagram illustrating the interleaved access of a common memory device in accordance with the present invention." *See* DeRoo, col. 3, ll. 31-33. That is, Figure 24 specifically illustrates the interleaved operation of the system during the time-multiplexed shared mode of operation disclosed by DeRoo: "during a shared mode, multiplexing of the common memory device 704 provides for interleaved access to the common memory device 704 by the CPU 702 and the SCP 706." *See id.*, col. 83, ll.28-31.

As can be seen from Figure 24, once shared mode operation begins, then so long as no entity requests exclusive access, CPU 702 and SCP 706 will continue to receive alternating access to common memory device 704 according to a fixed, predictable schedule. However, as discussed in the summary of DeRoo given above, regardless of whether the system is operating in shared mode or SCP-exclusive mode, CPU 702 can request exclusive access to common memory device 704 at any time "by placing the SCP 706 into a reset state by setting bit 0 of the flash interface control register FLASH\_CTRL." *See* DeRoo, col. 82, ll.2-4.

That is, during shared mode operation according to DeRoo, SCP 706 may periodically receive access to common memory device 704. However, the integrity or continuity of such access is in no way guaranteed by DeRoo; rather, the SCP's access can be preempted by CPU 702 at any time. Thus, the access of SCP 706 to common memory device 704 is clearly not certain or guaranteed to be free from interruption or interference by CPU 702—instead, such

---

<sup>2</sup> One of ordinary skill in the art would readily recognize that an assurance of safe access is not absolute, but rather is predicated on the existence of normal system operating conditions. Of course, defects, failures, or other exceptional circumstances may cause the system to malfunction.

interruption is by deliberate design how DeRoo permits CPU 702 to gain exclusive access to common memory device 704. Therefore, the mere fact that DeRoo's CPU SYSTEM ACCESS signal is in a low state at a given time cannot be said to correspond to a state in which SCP 706 is "assured safe access" to common memory device 704, for any meaningful interpretation of "assured safe access."

In fact, rather than providing "assured safe access" to SCP 706, the organization of DeRoo practically assures that any access by SCP 706 will be unsafe, because SCP 706 is not only susceptible to being preempted by CPU 702 at any time, but is in fact preempted with extreme prejudice, by being reset. That is, when CPU 702 demands exclusive access to common memory device 704, it does not merely interrupt any outstanding access by SCP 706 in such a way that SCP 706 could readily resume access when CPU 702 relinquishes its access. Rather, DeRoo states that "during periods when the CPU 702 has exclusive control . . . the SCP 706 is held in reset." *See* DeRoo, col. 81, ll.46-48 (emphasis added). One of ordinary skill would understand "held in reset" to mean that SCP 706 is held in an inactive, determinate state from which initial operation can proceed once the reset state is released, but which would erase the state of operations that were pending at the time reset was asserted. *See* 132 Declaration at ¶ 32. Thus, it appears that SCP 706 cannot rely on the continuity or consistency of any access to common memory device 704 that it initiates.

One practical example illustrating the lack of "assured safe access" for SCP 706 in DeRoo's system may be understood as follows. DeRoo states that one purpose of the common memory device shared by SCP 706 and CPU 702 is to store the firmware for SCP 706. *See* DeRoo, col. 36, l.9. One of ordinary skill would readily understand that the firmware of a microcontroller is code that controls the basic operations of the microcontroller, for example in order to initialize the microcontroller. *See* 132 Declaration at ¶ 7. In DeRoo's system, SCP 706 might begin fetching and executing firmware code from common memory device 704 using either SCP-exclusive or shared mode access, a process that may take a number of cycles depending on the length of the firmware routine to be fetched. However, as demonstrated above, at any time during SCP 706's fetching of firmware, CPU 702 may preempt SCP 706 by resetting

it in order to demand exclusive access for itself. This may lead to at least two possible negative outcomes.

First, because SCP 706 cannot guarantee its access to common memory device 704 for any length of time, it is possible that if CPU 702 repeatedly resets SCP 706, SCP 706 will never actually complete fetching a firmware routine (i.e., SCP 706 may be essentially deadlocked or prevented from making progress). *See* 132 Declaration at ¶ 34. More pathologically, because CPU 702 can gain exclusive access to common memory device 704 at any time, it is possible that CPU 702 could actually modify the firmware on which SCP 706 depends before or while SCP 706 is fetching that firmware, either accidentally (e.g., as a result of poorly-behaved code executing on CPU 702) or deliberately (e.g., in the case of malware). *See* 132 Declaration at ¶ 35. Thus, the operation of SCP 706 may be altered or corrupted due to the alteration or corruption of its firmware. Because of the basic organization of DeRoo's system (i.e., DeRoo's failure to provide "assured safe access" for SCP 706), when DeRoo's system is operating according to the protocol it specifically provides, DeRoo's system cannot prevent these outcomes of deadlock or corruption from occurring. *See* 132 Declaration at ¶¶ 36-37.

3. DeRoo fails to disclose that a microcontroller is configured to "hold the system in the first state."

In rejecting claim 1, the Office Action alleges that DeRoo's address latch enable N\_SCPALE corresponds to the feature "hold the system in the first state." *See* Office Action at 5. Patent Owner respectfully disagrees for at least the following reasons.

One of ordinary skill in the art would understand that, even under BRI, for a device to "hold" a system in a particular state, the device would persistently control the system to preserve that particular state until the device released such control and allowed the system state to transition to a different state. *See* 132 Declaration at ¶ 39. This is consistent with the ordinary meaning of the word "hold" that is relevant in this context, which is "to maintain (a certain condition, situation, or course of action) without change." *See* Merriam-Webster Online Dictionary, <http://www.merriam-webster.com/dictionary/hold>.



DeRoo's address latch enable signal N\_SCPALE does not "hold" DeRoo's system in a particular state, as one of ordinary skill would understand that term. DeRoo states that during shared mode operation, the address latch enable initiates an SCP access window: "[A]t the falling edge of SCP address latch enable SCPALE, indicating that the SCP address is valid, memory control is given to the SCP 706." *See* DeRoo, col. 83, ll.45-48. However, SCP 706 does not persistently maintain such control. Rather, DeRoo states that after SCPALE transitions, access is "switched to the CPU 702 after two rising and one falling edges of the SCP clock SCPCLK are detected." *See id.*, ll.55-57. That is, assertion of SCPALE initially gives SCP 706 memory access, but only temporarily: SCP 706 will automatically and unconditionally lose access to common memory device 704 after "two rising and one falling edges of the SCP clock."

Thus, one of ordinary skill would understand that DeRoo's address latch enable signal may be sufficient to initiate a state of access to common memory device 704 from SCP 706, but is insufficient to hold such a state (much less a "first state" in which SCP 706 is "assured safe access," as mentioned below). *See* 132 Declaration at ¶ 40. As mentioned in the overview of DeRoo in Section III, *supra*, this is further evidenced by DeRoo's Verilog source code appendix that describes the functionality of HUI 700. In the source code, DeRoo specifies the generation of the signal "flashSCP," in which the signal "flashSCP" directly determines which entity of SCP 706 or CPU 702 has access to common memory device 704. The signal "flashSCP" is set by the falling edge of signal "SCPALE" and is unconditionally cleared by the rising edge of signal "clr\_flashSCP," which is itself merely a version of the "flashSCP" signal that has been passed through a fixed delay chain. *See* DeRoo, cols. 129, 131; *see also* 132 Declaration at ¶ 41.

Leaving aside the question of the role of SCPALE, Patent Owner has already demonstrated that given DeRoo's basic system organization, SCP 706 is not capable of holding any particular system state. As discussed at length above, SCP 706 is subject to being reset by CPU 702 at any time. Accordingly, even if SCP 706 were to attempt to persistently preserve a particular system state, it would be unable to do so, because as soon as CPU 702 forced SCP 706 into reset, SCP 706 would no longer be able to preserve that system state. *See* 132 Declaration at ¶ 42.

4. DeRoo fails to disclose that a microcontroller is configured to “release the system from the first state.”

In rejecting claim 1, the Office Action alleges that DeRoo discloses the aspect of claim 1 wherein a microcontroller is configured to “release the system from the first state.” *See* Office Action at 5. However, the evidentiary basis of this allegation is not clear. The Office Action cites to FIG. 20 as well as several passages from DeRoo that do not appear to have anything to do with the feature of “release the system from the first state” for which they are offered. Patent Owner addresses these various citations in the Office Action as follows:

- The quotation from col. 36, ll.30-32 refers to the HUI “providing an interface to enable both the SCP and the CPU to share a common memory device.” Mere description that devices are shared does not disclose the particular state transition that is recited.
- The quotation from col. 36, ll.56-60 relates to how reads and writes by CPU 702 to HUI 700 are controlled. However, how CPU 702 (which the Office Action maps to the recited “processor”) interacts HUI 700 is not relevant to how SCP 706 (which the Office Action maps to the recited “microcontroller”) behaves.
- The reference to DeRoo’s control 720 “for accessing the common memory device 706 from either the CPU register file 712 or the SCP register file 714” is likewise irrelevant to the operation of SCP 706; it refers instead to the operation of HUI 700, which is a distinct ASIC from both CPU 702 and SCP 706.

The Office Action also cites to the portion of DeRoo describing the various control signals used by SCP 706 to communicate with common memory device 704, including the N\_SCPALE signal. However, as was demonstrated in the preceding section, although the N\_SCPALE signal serves to initiate a period of access by SCP 706 to common memory device 704 during shared mode operation, this access period is unconditionally terminated by HUI 700 after a fixed number of clock edges are detected. That is, SCP 706 does not “release” its control over access once the access period terminates; rather, HUI 700 forcibly terminates the access

period, regardless of the cooperation of SCP 706, so that access can be “switched to the CPU 702.” *See* DeRoo, col. 83, ll.55-57; *see also* 132 Declaration at ¶ 43.

More generally, as noted in the preceding section, SCP 706 is not capable of holding any particular system state, because SCP 706 is subject to reset at any time by CPU 702. Even when SCP 706 operates in SCP-exclusive mode, DeRoo does not disclose that SCP 706 deliberately “releases” or relinquishes its access of its own volitional control when CPU 702 requests exclusive access. Rather, as noted above, CPU 702 takes control in this circumstance “by forcing SCP 706 into reset.” *See* DeRoo, col. 81, ll.64-66; *see also* 132 Declaration at ¶ 43.

Because a device’s ability to “hold” a state is a logical prerequisite to the device’s ability to “release” a state (i.e., one cannot “release” something one does not “hold”) and because SCP 706 cannot assert sufficient control to “hold” a particular system state, it is illogical to suggest that SCP 706 could “release” a particular system state. *See* 132 Declaration at ¶ 44.

5. Contrary to the Office Action’s position, DeRoo does not disclose that a microcontroller is configured to perform certain actions *in response to* a change in state of the CPU SYSTEM ACCESS signal.

Claim 1 specifically recites a microcontroller that “is configured to[,] in response to a change in system state to a first state . . . hold the system in the first state and switch access to the non-volatile memory from the processor to the microcontroller.” This language is causal. That is, according to the plain language of the claim, the microcontroller is configured such that the “change in system state to a first state” causes the microcontroller to “hold the system in the first state” and “switch access” in response.

In rejecting claim 1, the Office Action alleges that the “change in system state to a first state” corresponds to a low-going transition on the CPU system access signal. *See* Office Action at 5. However, DeRoo illustrates CPU system access signal 2020 as a signal that is strictly internal to HUI 700. *See* DeRoo, Figure 23 (illustrating CPU ACCESS 2020 as a signal generated by control 720 that is routed only to address mux 716 and data mux and latch 718). DeRoo’s Verilog source code confirms that CPU system access signal 2020 (referred to as

SysAccess in the code; *see* DeRoo, col. 82, l.19) is internal to HUI 700. *See* DeRoo, col. 131, 133. However, if CPU system access signal 2020 is internal to HUI 700, and thus not an input to the microcontroller that is SCP 706, then a change in state on CPU system access signal 2020 cannot cause SCP 706 to perform any particular function.

That is, if microcontroller does not receive state information, it cannot act in response to a change in that state information. Because SCP 706 does not receive the state information that the Office Action alleges corresponds to the recited “first state,” SCP 706 cannot cause actions to occur in response to changes in this state information. Accordingly, the Office Action fails to establish that DeRoo has met this aspect of claim 1.

6. Because DeRoo fails to disclose or suggest numerous features of claim 1, DeRoo fails to support the rejection of claim 1.

The legal standard for establishing anticipation is strict. “[U]nless a reference discloses within the four corners of the document not only all of the limitations claimed but also all of the limitations arranged or combined in the same way as recited in the claim, it cannot be said to prove prior invention of the thing claimed and, thus, cannot anticipate under 35 U.S.C. § 102.” *Net MoneyIN, Inc. v. VeriSign, Inc.*, 545 F.3d 1359, 1371 (Fed. Cir. 2008). Anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 U.S.P.Q. 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 U.S.P.Q.2d 1913, 1920 (Fed. Cir. 1989).

As demonstrated above, DeRoo fails to disclose or suggest numerous features of claim 1. Therefore, for at least the reasons discussed above, DeRoo fails to render claim 1 anticipated or obvious.

#### B. CLAIM 2:

Claim 2 recites “wherein the change in system state to the first state is performed by software.” In rejecting claim 2, the Office Action alleges that DeRoo’s passage at col. 36, ll.42-44 discloses this feature. *See* Office Action at 6. The quoted passage of DeRoo states that “the HUI 700 allows the SCP 706 to communicate with the CPU 702 and the common memory device 704 by way of an SCP register file 714.”

DeRoo does not explicitly describe the use of software with respect to SCP register file 714. Patent Owner submits that the existence of a register file to facilitate communication between SCP 706, CPU 702, and common memory device 704 does not necessitate that the register file is manipulated by or even visible to software. It is entirely conceivable that such a register file could be controlled through hardware alone.

Moreover, as noted above with respect to claim 1, DeRoo fails to disclose the recited “first state,” and so cannot disclose how a change to that first state is achieved. However, even assuming *arguendo* that the Office Action’s position regarding the “first state” were correct (i.e., that the “first state” corresponds to a low level of DeRoo’s CPU SYSTEM ACCESS signal), DeRoo nevertheless lacks sufficient disclosure to establish that such software would be performing “the change in the system state to the first state.” As noted above (*see, e.g.*, Section IV.A.5, *supra*), DeRoo illustrates and describes the CPU SYSTEM ACCESS signal as a hardware-generated signal. In describing the operation of SCP 706 and CPU 702 with respect to sharing access to common memory device 704, DeRoo discusses how these devices interact at the pin or interface level, but does not discuss in any significant detail how software internal SCP 706 or CPU 702 causes pin-level behavior to occur. Consequently, the Office Action has not established evidence that in DeRoo, software is responsible for manipulating the CPU SYSTEM ACCESS signal, much less the “first state” recited in claim 2.

C. CLAIM 4:

Claim 4 is patentable over DeRoo for at least the reasons given above for claim 1.

D. CLAIM 5:

Claim 5 is patentable over DeRoo for at least the reasons given above for claim 1.

E. CLAIM 6:

Claim 6 is a method claim having numerous features that are similar to features of claim 1. In rejecting claim 6, the Office Action relies on a virtually identical mapping to portions of DeRoo that was employed in rejecting claim 1. *See* Office Action at 6-7. Patent Owner submits that the arguments given above with respect to claim 1 apply with equal force with respect to similar features of claim 6.

More specifically, claim 6 recites “a first state wherein the microcontroller is assured safe access to the non-volatile memory.” As set forth in detail in Section IV.A.2, *supra*, DeRoo’s system allows SCP 706 to be preempted at any time by CPU 702, regardless of whether DeRoo’s system is operating in shared mode or SCP-exclusive mode. As a result, SCP 706 is never “assured safe access” to DeRoo’s common memory device 704, because it cannot be assured access that occurs without interference of CPU 702. Because of the fundamental lack of “assured safe access” for SCP 706 in DeRoo’s system, it is possible that CPU 702 could deadlock SCP 706 by continually resetting it, never allowing it to complete fetching a needed firmware routine. It is further possible that CPU 702 could corrupt SCP 706’s firmware in a manner SCP 706 has no way to prevent.

Claim 6 additionally recites “holding the system in the first state.” As with claim 1, the Office Action alleges that DeRoo’s address latch enable signal N\_SCPALE accomplishes this feature. However, as set forth in detail in Section IV.A.3, *supra*, although N\_SCPALE is used to initiate a state of access to common memory device 704 from SCP 706, it cannot hold such a state, because in the shared mode of operation in which N\_SCPALE is relevant, HUI 700 unconditionally terminates SCP 706’s access after a fixed and determinate period of time. More generally, because SCP 706 can be reset by CPU 702 at any time as noted above, DeRoo’s system cannot be understood to perform “holding the system in the first state,” where the “first state” entails that the microcontroller is “assured safe access.”

Claim 6 additionally recites “releasing the system from the first state.” As set forth in detail in Section IV.A.4, *supra*, the evidentiary basis in the Office Action for the allegation that DeRoo discloses this feature is unclear. DeRoo’s address latch enable signal N\_SCPALE, which

is cited along with other passages of questionable relevance, was demonstrated above to be involved with initiating an access period by SCP 706 during shared mode operation, but is not involved in terminating or otherwise “releasing” the access period. More generally, because SCP 706 is not capable of “holding” any particular system state, it is not logically capable of “releasing” such a state.

Claim 6 further recites “in response to a change in system state to a first state . . . holding the system in the first state.” As set forth in detail in Section IV.A.5, *supra*, this claim language requires that the change in system state cause the “holding.” However, as discussed above, the CPU system access signal 2020 that is alleged to correspond to the “first state” does not appear to be capable of causing the resultant behavior in the manner recited in claim 6.

F. CLAIM 7:

Claim 7 recites features that are similar to claim 2. Patent Owner submits that claim 7 is patentable over DeRoo for at least the reasons given above for claim 2 as well as claims 1 and 6.

G. CLAIM 9:

Claim 9 is patentable over DeRoo for at least the reasons given above for claims 1 and 6.

H. CLAIM 10:

Claim 10 is patentable over DeRoo for at least the reasons given above for claims 1 and 6.

Rejections under 35 U.S.C. § 103(a):

I. CLAIM 3:

Claim 3 recites that the first state comprises one or more of “a power on reset (POR) state, a system reset state, or a sleep state.” In rejecting claim 3, the Office Action acknowledges that DeRoo does not explicitly teach that the first state comprises any of these states, but alleges that “powering on, resetting and sleep mode were well known initial states for memory system

operation.” See Office Action at 8. From this, the Office Action concludes that it would have been obvious to one of ordinary skill in the art to utilize one or more of the recited states within DeRoo’s system. Patent Owner respectfully disagrees for at least the following reasons.

First, the assertion that “powering on, resetting and sleep mode were well known initial states” is offered without any evidentiary support whatsoever. Given that the Office Action has acknowledged the absence of such features from DeRoo, Patent Owner respectfully requests that evidence beyond simple conjecture be made of record to substantiate the “well known” status of these states.

Second, in rejecting claim 1, the Office Action alleges that the “first state” corresponds to a low state of the CPU SYSTEM ACCESS signal. As discussed above with respect to claim 1 (*see, e.g.,* Section IV.A.5, *supra*), the CPU SYSTEM ACCESS signal is a signal derived by HUI 700 to coordinate which of SCP 706 and CPU 702 has access to common memory device 704 when the system is operating in shared mode. The Office Action fails to explain how, given DeRoo’s description of the CPU SYSTEM ACCESS signal, such a signal could be modified to incorporate any of the various states recited in claim 3. That is, it is insufficient to merely allege that DeRoo could be modified in the abstract to incorporate any of the states of claim 3 when claim 3 specifically limits the “first state” that is recited in claim 1.

Accordingly, Patent Owner submits that the Office Action’s evidence and reasoning is insufficient to support a *prima facie* obviousness rejection of claim 3.

J. CLAIM 8:

Claim 8 recites features that are similar to claim 3. Patent Owner submits that claim 8 is patentable over DeRoo for at least the reasons given above for claim 3 as well as claims 1 and 6.

K. CLAIM 11:

Claim 11 is a system claim that recites a non-volatile memory, a processor, and a microcontroller. Among other features, claim 11 recites that the microcontroller is configured to, “in response to a power on reset, hold a system reset signal in a reset state, thereby prohibiting



the processor from accessing the non-volatile memory.” Moreover, according to claim 11, the microcontroller is further configured to, “while the system reset signal is held in the reset state, fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller” and “after the program instructions or data have been loaded, permit the processor in the system access to the non-volatile memory, and release the system reset signal from the reset state.”

In rejecting claim 11 under 35 U.S.C. § 103(a), the Office Action acknowledges that DeRoo “does not explicitly teach” any of the features of claim 11 quoted above. *See* Office Action at 9. The Office Action further states that DeRoo does disclose “(1) holding a processor in a *wait* state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so” and “(2) holding the microcontroller in a reset state to prohibit the microcontroller from accessing the non-volatile memory, while allowing the processor to do so.” *See id.*

From this, the Office Action alleges that

it would have been obvious to one having ordinary skill in the art at the time the invention was made to hold the processor in the system of DeRoo in a reset state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller exclusive access to the non-volatile memory until the reset signal is released, since (1) DeRoo specifically taught the use of a reset state to prevent an element from accessing the non-volatile memory; namely, holding the microcontroller in a reset state to prohibit the microcontroller from accessing the non-volatile memory, while allowing the processor to do so, (2) DeRoo specifically taught the use of a state to prohibit the *processor* from accessing the non-volatile memory, while allowing the microcontroller exclusive access to the non-volatile memory, namely, holding a processor in a wait state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so and (3) it may have been advantageous to take make use of any idle time during the wait state of the processor by performing a reset operation in the processor to run new updates, etc.

*See* Office Action at 10.

Patent Owner respectfully disagrees with the Office Action’s position concerning the alleged obviousness of claim 11 for at least the following reasons.

1. The proposed modification would impermissibly change a fundamental principle of operation of DeRoo and would impermissibly render DeRoo unsatisfactory for its intended purpose.

The Office Action is correct in noting that DeRoo teaches the use of reset to prohibit the microcontroller from accessing the non-volatile memory. However, the Office Action errs in assuming that SCP 706 and CPU 702 could reasonably be modified to use a symmetrical reset scheme instead of the asymmetrical priority scheme that DeRoo actually discloses, because such a modification would impermissibly change a fundamental principle of operation of DeRoo and would impermissibly render DeRoo unsatisfactory for its intended purpose.

One of ordinary skill in the art would recognize that in any system in which multiple requesting devices may contend for access to a shared resource that cannot concurrently service the multiple requesting devices, some sort of conflict-resolution scheme is necessary for deciding which requesting device will succeed when there are concurrent (and thus conflicting) requests. *See* 132 Declaration at ¶ 48. For example, a system architect may choose to implement a fixed priority ordering system in which a higher-priority requesting device always receive access over a lower-priority requesting device when a conflict arises. *See* 132 Declaration at ¶ 49. Alternatively, a wide variety of more sophisticated arbitration schemes may be employed that actively decide which of several requesting devices will receive access when there is a conflict: for example, the winner may be chosen randomly/pseudorandomly, or by taking into account historical usage patterns of the various requesting devices (e.g., in order to ensure that one requesting device does not monopolize the shared resource at the expense of other devices). *See* 132 Declaration at ¶ 50.

One of ordinary skill in the art would further recognize that the choice of conflict-resolution scheme for a shared-resource system is a fundamental architectural decision that has implications for both hardware and software design. *See* 132 Declaration at ¶¶ 51-53. Specifically, once the decision to implement a particular conflict-resolution scheme is made, various system components and software may be designed to specifically rely on that conflict-resolution scheme, and may not operate efficiently (or even correctly) if used in a system that relies on a different conflict-resolution scheme. *See id.*

The fundamental implications of the choice of conflict-resolution scheme on the design of system components may be illustrated by the following example. Assume that in a fixed priority system where a processor always has higher priority than a microcontroller with respect to a shared memory, software executing on the processor is designed with this assumption in mind—i.e., that a process that needs access to the shared memory will never be delayed on account of the microcontroller. If such a process were to execute in a system that did not implement fixed priority access—e.g., if it were deployed in a system that arbitrated between the processor and the microcontroller in a round-robin fashion, or according to some other variable scheme—the basic assumption under which the process was designed is no longer correct. If the process depends on the assumption, it may operate incorrectly when the assumption is violated. For example, if the process is performing time-critical operations and assumes no delay will occur because of the microcontroller, then if a delay does occur, the process may miss a critical deadline. As a result, the system may malfunction. *See* 132 Declaration at ¶¶ 54-55.

Turning now specifically to DeRoo, it is apparent that DeRoo's system is fundamentally organized around the principle that CPU 702 is the highest-priority device with respect to common memory device 704. This is true even in the event that SCP 706 has previously placed CPU 702 in a wait state in order to gain exclusive access to common memory device 704, because DeRoo specifically states that “[d]uring conditions when the SCP has exclusive access of the common memory device 704, the CPU 702 can gain control by forcing the SCP 706 into reset.” *See* DeRoo, col. 81, ll.64-66.

Thus, one of ordinary skill in the art would recognize that DeRoo implements a fixed priority scheme in which CPU 702 unconditionally holds higher priority than SCP 706 with respect to accessing common memory device 704. *See* 132 Declaration at ¶ 56. One of ordinary skill in the art would further recognize that, as noted above, the choice of a fixed priority scheme in DeRoo reflects a basic architectural decision concerning system organization that gives rise to a fundamental principle of operation on which other system elements will rely for their correct operation. *See id.* For example, software executing on CPU 702 or other system hardware working in cooperation with CPU 702 may reasonably expect to rely on the fact that CPU 702 will have access to common memory device 704 at its discretion, without negotiation or delay on

the part of SCP 706.

From this, one of ordinary skill in the art would appreciate that altering DeRoo such that CPU 702 would no longer unconditionally hold higher priority than SCP 706 would substantially alter a fundamental operating principle of DeRoo. *See* 132 Declaration at ¶ 57, 59. Moreover, one of ordinary skill in the art would recognize that modifying DeRoo in the manner proposed in the Office Action—i.e., such that SCP 706 could reset CPU 702 in the same manner that CPU 702 can reset SCP 706—would entail that CPU 702 no longer unconditionally hold higher priority than SCP 706. *See* 132 Declaration at ¶ 58. This is reflected in the fact that as a result of such a modification, CPU 702 no longer can expect to access common memory device 704 at its discretion, but instead would be subject to possible delay in the event that it is reset by SCP 706.

However, it is a basic tenet of the law of obviousness that “[i]f the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious.” *See* M.P.E.P. 2143.01.VI (internal citations omitted). Because the Office Action’s proposal to modify DeRoo’s system such that SCP 706 could reset CPU 702 would change the principle of operation of DeRoo, such a modification cannot support the Office Action’s allegation that claim 11 would have been obvious to one of ordinary skill in the art in view of DeRoo.

Moreover, there are several sound technical reasons why DeRoo uses reset asymmetrically (i.e., such that CPU 702 can reset SCP 706 but not vice versa) as opposed to the symmetrical scheme proposed by the Office Action. First, If SCP 706 had the proposed ability to reset CPU 702, then it would be possible for both CPU 702 and SCP 706 to concurrently attempt to reset each other. This creates a possibility for a serious failure mode to arise: if the two devices were to succeed in holding each other in a reset state, a deadlock condition could occur in which neither device is able to proceed. *See* 132 Declaration at ¶ 61. As a result of such a deadlock, DeRoo’s system would hang, likely requiring a power cycle in order to restore normal operation. *See id.*

Second, DeRoo specifically indicates that the intended purpose of SCP 706 is for use in a

personal computer in which CPU 702 corresponds to the general purpose processor of the personal computer:

As mentioned above, known IBM type AT PC-compatible personal computers utilize an SCP 706 with an internal ROM for program storage. It is also known that the BIOS is stored in a separate memory device. The HUI 700 allows both the CPU 702 and the SCP 706 to share a common memory device 704.

*See* DeRoo, col. 81, ll.37-42; *see also* col. 4, ll.24-26 (stating that “known newer type PC/AT machines” use Intel x86-type processors as the CPU). One of ordinary skill in the art would understand that a personal computer with CPU 702 at its core would be expected to run an operating system as well as user applications. *See* 132 Declaration at ¶ 62.

As noted above with respect to claim 1, one of ordinary skill would understand the manner in which DeRoo holds SCP 706 “in reset” to mean that SCP 706 is held in an inactive, determinate state from which initial operation can proceed once the reset state is released, but which would erase the state of operations that were pending at the time reset was asserted. The Office Action proposes to give SCP 706 the ability to arbitrarily reset CPU 702 in the same manner in which CPU 702 can reset SCP 706, instead of merely requesting CPU 702 to wait as DeRoo describes. But if SCP 706 had such a capability with respect to CPU 702, then at any given time, SCP 706 could erase the current operating state of CPU 702, interrupting any pending activities of the operating system or user applications executing on CPU 702. *See* 132 Declaration at ¶ 63. In effect, giving SCP 706 such a reset capability over CPU 702 would allow SCP 706 to reboot the computer at any time, without notice to the user.

Thus, the Office Action’s proposed modifications to DeRoo create the possibility of significant negative consequences: system hangs arising from deadlock conditions, and the potential for arbitrary system reboots arising from resetting of CPU 702. Both of these consequences are undesirable for a computer system: at a minimum, they may cause a user to lose work, and in the worst case they may render the system effectively unusable. As a result, the Office Action’s proposed modifications render DeRoo unsatisfactory for its intended purpose of use within a personal computer system. *See* 132 Declaration at ¶¶ 64-65.

It is a further principle of the law of obviousness that “[i]f [a] proposed modification

would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification.” See M.P.E.P. 2143.01.V (internal citations omitted). Because the Office Action’s proposal to modify DeRoo’s system such that SCP 706 could reset CPU 702 would render DeRoo unsatisfactory for its intended purpose of use within a personal computer system, such a modification cannot support the Office Action’s allegation that claim 11 would have been obvious to one of ordinary skill in the art in view of DeRoo.

In summary, the Office Action’s proposal for modifying DeRoo cannot be said to render claim 11 obvious to one of ordinary skill in the art for several distinct reasons: (1) it would impermissibly alter the fundamental principle of operation of DeRoo, and (2) it would impermissibly render DeRoo unsuitable for its intended use in a personal computer system in several ways.

2. Contrary to the Office Action’s assertion, DeRoo fails to disclose “prohibiting the processor from accessing the non-volatile memory,” leaving some features of claim 11 unaccounted for.

In rejecting claim 11, the Office Action alleges that DeRoo discloses “holding a processor in a *wait* state to prohibit the processor from accessing the non-volatile memory.” See Office Action at 9. Although Patent Owner agrees that DeRoo describes how SCP 706 can place CPU 702 in a wait state in order to enter the SCP-exclusive mode of operation, Patent Owner disagrees that this amounts to prohibiting CPU 702 from accessing common memory device 704.

Patent Owner submits that, under the BRI standard for claim construction discussed above, the meaning of “prohibiting” is governed by the ordinary meaning of this term consistent with its usage in the specification. Specifically, the ordinary meaning of the infinitive “to prohibit” that is consistent with the usage of “prohibiting” in the ’576 patent is “to prevent from doing something; preclude.” See Merriam-Webster Online Dictionary, <http://www.merriam-webster.com/dictionary/prohibit>.

As thoroughly demonstrated above with respect to claim 1, CPU 702 is not in fact

prevented or precluded from accessing common memory device 704 when SCP 706 has exclusive access, because even “[d]uring conditions when the SCP has exclusive access of the common memory device 704, the CPU 702 can gain control by forcing the SCP 706 into reset.” *See* DeRoo, col. 81, ll.64-66. That is, even when SCP 706 has exclusive access, CPU 702 may unconditionally regain control over common memory device 704. This is also supported by DeRoo’s description of the control register through which CPU 702 asserts its demand for exclusive access: in describing the CPUEXCL flag of the FLASH\_CTRL register, DeRoo states that this flag indicates that “SCP has requested exclusive access to flash memory for a write and CPU is disallowed from reading and writing the FLASH-this bit can be cleared immediately if a ‘1’ is written to SCPreset.” *See id.*, col. 49, Table XXXI (emphasis added); *see also* 132 Declaration at 16-22.

Thus, even under a broadest reasonable interpretation of the term “prohibiting,” one of ordinary skill in the art would not consider DeRoo’s placing of CPU 702 into a wait state to amount to “prohibiting” CPU 702 from accessing common memory device 704. At most, DeRoo’s placing of CPU 702 into a wait state amounts to a temporary delay, if the request is honored by CPU 702 at all.

Thus, as shown above, the Office Action has failed to establish a prima facie case of obviousness of claim 11, in that the proposed modifications to DeRoo are impermissible to establish obviousness. Moreover, DeRoo fails to disclose that a processor is “prohibited” from accessing a non-volatile memory, leaving this aspect of claim 11 unaccounted for.

L. CLAIM 12:

Claim 12 is patentable over DeRoo for at least the reasons given above for claim 11.

M. CLAIM 13:

Claim 13 is patentable over DeRoo for at least the reasons given above for claim 11.

N. CLAIM 14:

Claim 14 is patentable over DeRoo for at least the reasons given above for claim 11.

O. CLAIM 15:

Claim 15 is patentable over DeRoo for at least the reasons given above for claim 11.

P. CLAIM 16:

Claim 16 is patentable over DeRoo for at least the reasons given above for claim 11.

Q. CLAIM 17:

Claim 17 is a method claim having numerous features that are similar to features of claim 11. In rejecting claim 17, the Office Action relies on a virtually identical rationale as was given with respect to claim 11. *See* Office Action at 11-13. Patent Owner submits that the arguments given above with respect to claim 11 apply with equal force with respect to similar features of claim 17.

More specifically, claim 17 recites “in response to a power on reset, holding a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory.” In rejecting claim 17, the Office Action acknowledges that DeRoo does not disclose this feature, but alleges that one of ordinary skill in the art could have modified DeRoo to allow SCP 706 to hold CPU 702 in a reset state, rather than a wait state. *See* Office Action at 12. As set forth in detail in Section IV.K.1, *supra*, such a modification would be impermissible as the basis for an obviousness rejection, because it would both change a fundamental operating principle of DeRoo, and would render DeRoo’s computer system unsuitable for its intended purpose. Moreover, as demonstrated in Section IV.K. 2, *supra*, DeRoo’s “wait state” does not in fact result in “prohibiting” a processor from accessing a non-volatile memory, leaving this aspect of claim 17 unaccounted for.



R. CLAIM 18:

Claim 18 is patentable over DeRoo for at least the reasons given above for claims 11 and 17.

S. CLAIM 19:

Claim 19 is patentable over DeRoo for at least the reasons given above for claims 11 and 17.

T. CLAIM 20:

Claim 20 is patentable over DeRoo for at least the reasons given above for claims 11 and 17.

Additionally, claim 20 recites that “the microcontroller is coupled to the processor via a system reset bus, and wherein the microcontroller is operable to assert the system reset signal over the system reset bus.” In rejecting claim 20, the Office Action refers only to Figure 20 of DeRoo, alleging that this figure shows a “RESET line for SCP.” *See* Office Action at 13.

However, as is plainly shown in DeRoo’s Figure 20 by the arrow pointing towards SCP 706, the RESET line coupled to SCP 706 is an input to SCP 706. A device that receives a signal as an input cannot, by definition, assert that signal, because it does not drive that signal; rather it receives the signal. As is shown in DeRoo’s Figure 20, it is control block 720 that drives the RESET line of SCP 706 as the SCPRESET output of HUI 700. This is reflected in the Verilog source code provided by DeRoo, which declares signal “SCPreset” as an output of module “security.” *See* DeRoo, col. 173.

Moreover, by its own terms, a “system reset signal” should have a reset effect on a system, and not merely on an isolated element of a system. As shown in Figure 20, the RESET input to SCP 706 is coupled only to SCP 706, and not to other elements of DeRoo’s system. Thus, this signal is at most a device-specific reset signal, and not a “system reset signal” as recited in claim 20.

U. CLAIM 21:

Claim 21 is patentable over DeRoo for at least the reasons given above for claims 11 and 17.

V. CLAIM 22:

Claim 22 is patentable over DeRoo for at least the reasons given above for claims 11 and 17.

W. CLAIM 23:

Claim 23 is patentable over DeRoo for at least the reasons given above for claims 11 and 17.

## **V. New Dependent Claims**

As noted above, new dependent claims 24-213 have been added. Patent Owner submits that each of the new dependent claims is patentable for at least the reasons given above with respect to the independent claims. Patent Owner notes that additional reasons for patentability apply to new claims 24-213 beyond those given with respect to the independent claims. However, because the patentability of the independent claims has been demonstrated, further discussion of the new dependent claims is unnecessary at this time.

## VI. Conclusion

In view of the foregoing remarks, Patent Owner respectfully requests removal of all pending rejections and confirmation of the patentability of issued claims 1-23 as well as newly added dependent claims 24-213. The Commissioner is authorized to charge any fees that may be required, or credit any overpayment, to Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C. Deposit Account No. 501505/5707-26200/JCH.

Also filed herewith are the following items:

- ☒ Declaration of Thomas M. Conte Under 37 C.F.R. § 1.132
- ☒ Certificate of Service

Respectfully submitted,

Date: April 23, 2012

By: /Anthony M. Petro/  
Anthony M. Petro  
Reg. No. 59,391

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.  
P. O. Box 398  
Austin, Texas 78767  
(512) 853-8800

**EX PARTE REEXAMINATION**

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

|                      |  |                                      |                            |
|----------------------|--|--------------------------------------|----------------------------|
| Reexamination        |  | §                                    |                            |
| Control. No.:        | 90/011,754   | §                                    | Atty. Dkt. No.: 5707-26200 |
| Inventor:            | Ian F. Harris, et al.  | §                                    | Examiner: Peikari, Behzad  |
| Request Filing Date: | June 21, 2011  | §                                    | Group/Art Unit: 3992       |
| U.S. Patent No.:     | 7,930,576  | §                                    | Conf. No.: 4071            |
| Title:               | SHARING NON-SHARABLE<br>DEVICES BETWEEN AN<br>EMBEDDED CONTROLLER<br>AND A PROCESSOR IN A<br>COMPUTER SYSTEM | §<br>§<br>§<br>§<br>§<br>§<br>§<br>§ |                            |

**DECLARATION OF THOMAS M. CONTE UNDER 37 C.F.R. § 1.132**

I, Thomas M. Conte, state as follows:

**I. Introduction**

1. I have been retained as an independent expert witness on behalf of Standard Microsystems Corporation ("SMSC"), the assignee of U.S. Patent No. 7,930,576 (the "'576 patent"). I did not contribute to the inventions described in the '576 patent.

2. I am a professor at the Georgia Institute of Technology ("Georgia Tech") with a joint appointment in the Schools of Computer Science and Electrical & Computer Engineering. I hold a Ph.D. degree in Electrical Engineering from the University of Illinois at Urbana-Champaign. I am an IEEE Fellow and a member of the Board of Governors of the IEEE Computer Society. I am also currently editor in chief of journal ACM Transactions on Architecture and Compiler Optimization, and am an associate editor of IEEE Transactions on Computers, and the IEEE Computer and IEEE Micro

magazines. A detailed account of my experience and other qualification is listed in my Curriculum Vitae attached as Exhibit 1.

## **II. Retention and Compensation**

3. I am compensated by SMSC at the rate of \$450 per hour, plus reimbursement for reasonable expenses incurred in the course of performing my services. My compensation is not contingent upon the outcome of this proceeding.

## **III. Basis of My Opinion and Materials Considered**

4. In preparation for this declaration, I have considered and relied on data or other documents identified herein. For example, I reviewed the '576 patent, the Office Action dated February 23, 2012 in the present reexamination, and the DeRoo reference cited in the Office Action.

5. My opinions are also based upon my education, training, research, knowledge, and experience in this field.

## **IV. Summary of My Opinion**

6. I have reviewed the Office Action of February 23, 2012, which includes a rejection of independent claims 1 and 6 of the '576 patent as being anticipated by DeRoo, and a rejection of independent claims 11 and 17 of the '576 patent as being obvious in view of DeRoo. It is my opinion that DeRoo does not anticipate claims 1 or 6, nor does DeRoo render claims 11 or 17 obvious. As detailed below, DeRoo fails to disclose a number of features of claims 1 and 6, including at least the "first state wherein the microcontroller is assured safe access to the non-volatile memory" feature. Moreover, the modification of DeRoo proposed by the Office Action in rejecting claims 11 and 17 would fundamentally alter the architecture of DeRoo and thus its core principle of operation, and would make DeRoo's system unsuitable for its intended use in a personal computer.

## **V. Detailed Discussion**

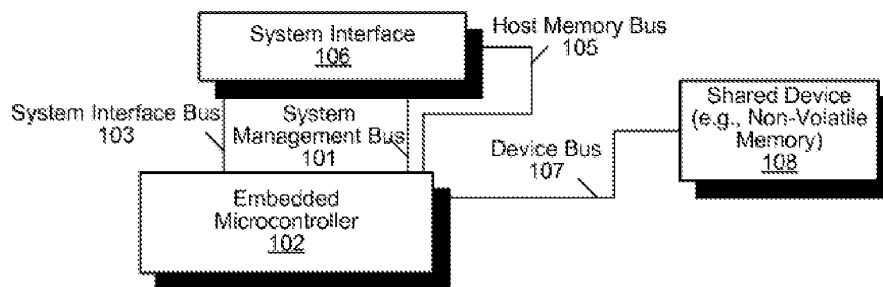
### *A. The '576 Patent*

7. The '576 patent is directed to techniques for permitting a device, such as a non-volatile memory, to be shared among multiple processors in a system, such as a central processing unit (CPU) and a microcontroller. Specifically, the '576 patent notes that each of multiple processors in a system may rely on its own firmware for managing certain processor functions, such as initialization and power management, among other functions. One of ordinary skill in the art would understand that firmware, as that term is used in the '576 patent, refers to executable code (and, possibly, associated data structures) that controls basic operations of a device, such as a microcontroller. Firmware may perform specialized functions, such as device initialization at startup, and/or more general functions, such as post-initialization operations a device may carry out during ordinary operation.

8. Historically, a system might have included separate non-volatile memories (e.g., as separate devices) in which to store different firmware for different devices. Alternatively, individual processors in the system might include embedded non-volatile storage manufactured on the same die as the processor. However, both of these techniques increase device and system cost, either due to requiring multiple discrete devices or due to the higher costs of fabricating a processor using a process suitable for embedded non-volatile memory. In another approach, individual processors might arbitrate for access to a shared device that stores firmware for the multiple processors. However, designing an arbitration circuit adds its own costs and complexities to system design, and depending on the arbitration scheme that is chosen, arbitration may degrade overall system performance, for example by introducing additional delay in resolving access contention by multiple processors.

9. The '576 patent proposes a system in which a microcontroller and a processor may share a non-volatile memory, thus overcoming the costs of providing separate or embedded non-volatile memories for the microcontroller and processor. However, the system of the '576 patent further overcomes the need for arbitration, and its associated

costs, by providing the microcontroller with a system state (also referred to as a “first state”) in which it is assured safe access to the shared non-volatile memory. An example arrangement of the system elements described in the ’576 patent is illustrated in Figure 1B:



10. By virtue of the “assured safe access” that is provided by the first state, the microcontroller may access the non-volatile memory without concern for whether the microcontroller’s access will be preempted or otherwise interfered with by the processor. After the microcontroller completes its access to the non-volatile memory, it releases the system from the first state, thus allowing the processor to access the shared non-volatile memory. The ’576 patent notes that the “first state” may correspond to a reset state, such as a power-on reset, or to a system reset state or sleep state.

11. As an example of how the system shown above might operate, following an initial power-on reset, embedded microcontroller 102 may come out of reset first, while holding system interface 106 (and, by extension, processor(s) coupled via system interface 106) in reset. In this state, microcontroller 102 may be assured safe access to shared device 108, by virtue of the fact that other devices (such as processors) within the system are held in reset and so cannot issue operations to access shared device 108. Consequently, microcontroller 102 may access shared device 108 to retrieve whatever firmware it needs without concern for whether its access will be interrupted or corrupted. Once



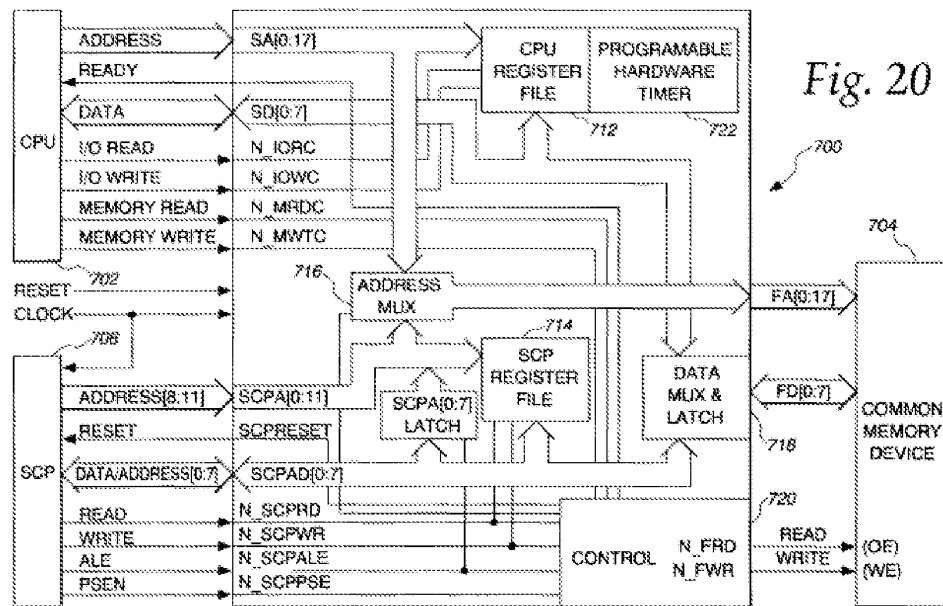
microcontroller 102 has completed its access, it may release other devices from reset so that they may access shared device 108.

12. One of ordinary skill would understand that for a microcontroller to be “assured safe access,” as that term is employed in the ’576 patent, the microcontroller must be guaranteed access to the shared memory in a manner that is free from interruption or interference from other processors in the system. More specifically, one of ordinary skill would understand that to be “assured safe access,” the microcontroller’s access cannot be arbitrarily preempted by another processor in such a way that the microcontroller’s operation could be adversely affected. If a microcontroller’s access to a shared device could be arbitrarily preempted by another processor, then the integrity of the microcontroller’s access could not be assured—that is, the access would not be “assured safe.” For example, such preemption by another device makes it possible for the other device to continually interrupt the microcontroller in a manner such that the microcontroller cannot complete its access, or for the other device to alter or corrupt data the microcontroller is attempting to access before the microcontroller has fully read such data from the shared device.

13. One of ordinary skill would further recognize that the ’576 patent discloses a system in which the microcontroller can preempt the processor, but in which the processor cannot arbitrarily preempt the microcontroller in such a way that would compromise the microcontroller’s “assured safe access.” That is, the ’576 patent discloses a system in which, by virtue of the basic organization of the system, the microcontroller’s access priority with respect to the shared memory is higher than the processor’s.

*B. DeRoo*

14. DeRoo is directed to a system that also provides a common memory device that can be shared by a processor and a microcontroller, albeit in a substantially different manner than the system of the ’576 patent. An example of DeRoo’s system is shown in Figure 20:



Specifically, DeRoo provides a central processing unit (CPU) 702 and a system control processor (SCP) 706, each of which is coupled to a human user interface (HUI) 700 through which they may access common memory device 704.

15. DeRoo provides three distinct modes of system operation with respect to common memory device 704: a mode in which CPU 702 has exclusive access to the common memory device, a mode in which SCP 706 has exclusive access to the common memory device, and a mode in which both CPU 702 and SCP 706 share access to the common memory device. DeRoo explains that exclusive access is required for either SCP 706 or CPU 702 to be able to write to common memory device 704. *See* DeRoo, col. 82, ll.21-22, 33-34. By contrast, in shared mode operation, common memory device 704 is configured for read-only access. *See id.*, ll.43-44.

16. DeRoo describes that CPU 702 can gain exclusive access to common memory device 704 by placing SCP 706 into a reset state. *See* DeRoo, col. 82, ll.1-3. DeRoo repeatedly notes that it is the act of being held in reset by CPU 702 that prevents SCP 706

from being able to access common memory device 704 when CPU 702 has exclusive access. *See id.*, col. 82, ll.29-31; col. 81, ll.46-49.

17. Moreover, DeRoo unambiguously states that the ability of CPU 702 to gain exclusive access to common memory device 704 is unconditional and independent of whether SCP 706 has exclusive access. DeRoo specifically states that “[d]uring conditions when the SCP has exclusive access of the common memory device 704, the CPU can gain control by forcing the SCP 706 into reset.” *See* DeRoo, col. 81, ll.64-66 (emphasis added).

18. By contrast, when SCP 706 attempts to gain exclusive control, it does not reset CPU 702, but instead places CPU 702 into a wait state by causing HUI 700 to deassert the IOCHRDY signal. *See* DeRoo, col. 81, ll.55-58. The IOCHRDY signal is defined by the standards for the Industry Standard Architecture/Extended Industry Standard Architecture (ISA/EISA) bus interface, which was commonly used for interfacing peripheral devices to computer systems based on the original IBM personal computer (PC) system architecture. As explained in the reference “ISA & EISA Theory and Operation,” by Edward Solari (1993) (“Solari”), relevant excerpts of which are attached as Exhibit 2, the function of the IOCHRDY signal is to “allow[] resources to indicate to the ISA or E-ISA bus master that additional cycle time is required.” *See* Solari at 159. That is, when IOCHRDY is inactive, the bus master (CPU 702, in the context of DeRoo) is requested to wait before accessing the ISA/EISA bus. However, as noted in the previous paragraph, CPU 702 is not required to honor SCP 706’s request for exclusive access, because DeRoo states that CPU 702 can force SCP 706 into reset even when SCP 706 has exclusive access. *See* DeRoo, col. 81, ll.64-66 (emphasis added).

19. In shared mode, DeRoo implements what is referred to as a “time based interleaving method” in which CPU 702 and SCP 706 are given access to common memory device 704 in an alternating fashion. *See* DeRoo, col. 81, l.51 & Figure 24. In this mode of operation, SCP 706 is given access to the shared memory when the SCP address latch enable signal is asserted, and after a fixed period of time (two rising edges

and one falling edge of the SCP clock), CPU 702 is given access. *See id.*, col. 83, ll.48-50, 53-57.

20. The amount of time that SCP 706 has access to the shared memory is controlled not by SCP 706, but by HUI 700, as is shown by the example Verilog implementation of HUI 700 DeRoo provides. Specifically, the signal “flashSCP” (which determines whether the SCP has access to the common memory device 704) is set by the falling edge of signal “SCPALE” and is cleared by the rising edge of signal “clr\_flashSCP,” which is itself merely a version of the “flashSCP” signal that has been passed through a fixed delay chain. *See* DeRoo, cols. 129, 131. Accordingly, one of ordinary skill would recognize that DeRoo’s shared mode is implemented as a form of time division multiplexing, in which the schedule that governs the amount of time a given device has access to the shared device is predictable and outside the control of the device.

21. Moreover, it is apparent from DeRoo’s description that when the system is operating in shared mode, CPU 702 may gain exclusive control of common memory device 704 by placing SCP 706 into reset, just as in the case when SCP 706 has exclusive control of common memory device 704. At the bottom of column 81, DeRoo describes the case where SCP 706 “has exclusive access of the common memory device 704,” stating that “CPU 702 can gain control by forcing the SCP 706 into reset.” At the top of column 82, DeRoo describes an “alternative[.]” case to the case where SCP 706 has exclusive access, and states that in this case, “CPU 702 can gain control by placing the SCP 706 into a reset state.” Because CPU 702 would not need to “gain control” if it already had exclusive control, and because DeRoo only describes three possible modes of operation (CPU-exclusive, SCP-exclusive, and shared), one of ordinary skill would understand that the alternative case DeRoo refers to at the top of column 82 is shared mode.

22. In summary, one of ordinary skill in the art would recognize that DeRoo discloses a system in which CPU 702 can preempt SCP 706’s access to common memory device 704 at any time, regardless of whether SCP 706 has exclusive access to the shared device, or whether CPU 702 and SCP 706 are sharing access in a time-division multiplexed

fashion. CPU 702 of DeRoo thus functions as a non-preemptible master device, whereas SCP 706 functions as a slave device that can request exclusive or shared access to common memory device 704, but will maintain such exclusive or shared access only so long as CPU 702 does not preempt SCP 706. That is, DeRoo discloses a system in which, by virtue of the basic organization of the system, a processor's (i.e., CPU 702's) access priority with respect to the shared memory is unconditionally higher than the microcontroller's (i.e., SCP 706's).

*C. One of ordinary skill would not consider DeRoo's SCP 706 and HUI 700 to correspond to a "microcontroller."*

23. In rejecting the independent claims, the Office Action alleges that the feature of DeRoo that corresponds to the claimed "microcontroller" is the combination of SCP 706 and HUI 700. *See* Office Action at 4, 6, 9, and 11. In my opinion, this position is incorrect, in that it is not consistent with the meaning of "microcontroller" as it would be understood by one of ordinary skill in the art, or as that term is used in both the '576 patent and in DeRoo.

24. "Microcontroller" is a highly generic term that can refer to a variety of devices having a variety of configurations. However, one of ordinary skill in the art would understand that in general usage, microcontrollers have a common characteristic: they refer to a single integrated circuit device. That is, they refer to a semiconductor device manufactured or fabricated as a single "chip" that is discretely packaged for integration into a larger system, e.g., a device to be controlled by the microcontroller.

25. This meaning of "microcontroller" is consistent with the '576 patent's usage of the term. For example, the '576 patent refers to the "microcontroller chip" when discussing the costs of embedding non-volatile memory within the microcontroller itself; such embedding requires the use of a flash memory manufacturing process in order to integrate the non-volatile memory "on the same die" as the microcontroller (i.e., as part of a single "chip" or integrated circuit). *See* '576 patent, col. 1, ll.54-56.

26. DeRoo also uses the term “microcontroller” consistent with its ordinary meaning of “single integrated circuit device.” Specifically, DeRoo states that SCP 706 is a microcontroller, and names the Intel 80C51 and 80C31 microcontrollers as examples of microcontrollers that may be used to implement SCP 706. *See* DeRoo, col. 36, ll.16-27. These devices are well known to be single-chip microcontrollers. *See, e.g.,* John Wharton, “An Introduction to the Intel® MCS-51™ Single-Chip Microcomputer Family,” Intel Application Note AP-69, at 2 (May 1980) (“Now three new high-performance single-chip microcomputers—the Intel® 8051, 8751, and 8031—extend the advantages of Integrated Electronics to whole new product areas.”) (emphasis added) (attached as Exhibit 3).

27. Moreover, DeRoo specifically describes HUI 700 as a single integrated circuit device in its own right that is distinct from SCP 706. DeRoo identifies HUI 700 as “an application specific integrated circuit (ASIC) which integrates the SCPI and MKI interfaces discussed above into a single device along with an interface for a common memory device.” *See* DeRoo, col. 36, ll.3-7 (emphasis added).

28. Thus, it is my expert opinion that one of ordinary skill would understand DeRoo’s SCP 706 and HUI 700 to be distinct integrated circuits, and therefore would not understand both of these devices taken together to constitute a “microcontroller,” particularly in view of DeRoo’s clear usage of the term “microcontroller” in its ordinary sense to refer to SCP 706 alone. Therefore, it is my expert opinion that the Office Action’s position that the recited “microcontroller” corresponds to both SCP 706 and HUI 700 is incorrect.

*D. DeRoo does not disclose “a first state wherein the microcontroller is assured safe access to the non-volatile memory.”*

29. Claims 1 and 6 of the ’576 patent recite “a first state wherein the microcontroller is assured safe access to the non-volatile memory.” It is my opinion that DeRoo does not provide any state in which SCP 706 is “assured safe access” to common memory device 704. Accordingly, it is my opinion that DeRoo does not disclose the “first state” recited in claims 1 and 6.

30. As noted above in paragraph 12, for a microcontroller to be “assured safe access,” the microcontroller must be guaranteed access to the shared memory in a manner that is free from interruption or interference from other processors in the system. More specifically, the notion of “assured safe access” precludes arbitrary preemption of the microcontroller’s access by another processor in such a way that the microcontroller could be adversely affected.

31. As noted above in paragraphs 16-22, DeRoo’s CPU 702 can arbitrarily preempt SCP 706 at any time. For example, regardless of whether SCP 706 has exclusive access to common memory device 704 or whether SCP 706 and CPU 702 are sharing access to common memory device 704 in a time-division-multiplexed fashion, CPU 702 can place SCP 706 into reset and thereby gain exclusive control of common memory device 704.

32. DeRoo states that SCP 706 is “held in reset” for the duration of “periods when the CPU 702 has exclusive control.” *See* DeRoo, col. 81, ll.46-48. One of ordinary skill in the art would understand a device’s being “held in reset” to mean that the device is persistently maintained in an inactive, determinate state from which the device’s initial operation could proceed once the state is released. However, the reset state would erase the state of operations that were pending at the time reset was asserted. That is, “reset” is commonly understood to denote a consistent, well-defined initial state of a device that can be invoked regardless of the device’s prior operational state. As such, after a device has been “held in reset” as discussed by DeRoo, it would not be able to resume operations that were in progress when reset was asserted, but would instead be forced to commence operating from the initial reset state.

33. Because DeRoo’s SCP 706 can be arbitrarily reset by CPU 702 at any time, SCP 706 cannot assume that any access it initiates to common memory device 704 will successfully complete, or that the data SCP 706 is attempting to access within common memory device 704 will not have been modified by CPU 702 during the course of SCP 706’s attempts to access such data. From this, several failure modes can possibly arise in DeRoo’s system.

34. First, DeRoo's system organization creates the possibility that SCP 706 will be deadlocked by CPU 702. For example, SCP 706 may attempt to read a firmware routine or other code or data from common memory device 704. However, before SCP 706 is able to complete its access, CPU 702 may preempt SCP 706 by resetting it. As noted in paragraph 32 above, because CPU 702 does not merely interrupt SCP 706 but resets it, SCP 706 would not be able to simply resume its access to common memory device 704 at the level of progress as of the time when CPU 702 preempted SCP 706. Rather, SCP 706 would need to begin its access anew. If CPU 702 were to repeatedly reset SCP 706 in this fashion, SCP 706 would effectively be prevented from completing its access to common memory device 704, and would thus be deadlocked (i.e., prevented from making progress). Put another way, given DeRoo's system organization, there is no assurance provided to SCP 706 that it will be able to complete any given read or write access to common memory device 704.

35. Second, DeRoo's system organization creates the possibility that CPU 702 can alter or corrupt data stored in common memory device 704 on which SCP 706 depends. For example, because CPU 702 can gain exclusive access (and thus write capability) to common memory device 704 at any time, CPU 702 may modify or overwrite SCP 706's firmware, either inadvertently, as a consequence of malfunctioning code executing on CPU 702, or maliciously, as a consequence of malware executing on CPU 702 that deliberately attempts to compromise the operation of SCP 706.

36. Because of the fundamental organization of DeRoo's system, which gives CPU 702 the unconditional ability to preempt SCP 706, DeRoo's system cannot guarantee against the possibility that failure modes like deadlock or corruption of SCP 706 operation will arise.

37. Because DeRoo's system allows for the unconditional preemption of SCP 706, and because such preemption can compromise the behavior of SCP 706, it is my expert opinion that DeRoo does not disclose or suggest "a first state wherein the microcontroller is assured safe access to the non-volatile memory," as recited in claims 1 and 6 of the '576 patent.



*E. DeRoo does not disclose that a microcontroller is configured to “hold the system in the first state” or “release the system from the first state.”*

38. Claims 1 and 6 of the '576 patent further recite that a microcontroller is configured to “hold the system in the first state” and “release the system from the first state.” In rejecting claims 1 and 6, the Office Action appears to allege that DeRoo’s address latch enable signal N\_SCPALE corresponds to the “hold the system in the first state” feature. *See* Office Action at 5, 6. The precise feature that the Office Action contends corresponds to the feature “release the system from the first state” is not clear; the address latch enable signal N\_SCPALE is cited again, along with a number of references to portions of DeRoo that simply appear to be irrelevant to the specific claim features for which they are cited. It is my opinion that neither the N\_SCPALE signal nor any other aspect of DeRoo related to the operation of SCP 706 discloses these particular claim features.

39. As noted above in paragraph 32 with respect to the phrase “held in reset,” one of ordinary skill would understand that for a device to “hold” a state, the device would be required to persistently maintain that state until that device releases the state, allowing it to change to a different state. That is, one of ordinary skill would recognize that “holding” a state involves concepts of agency and control on the part of the device that performs the holding, such that through the control of the device, the state is preserved or maintained over time, as opposed to merely being initiated but then allowed to change. Viewed another way, if a device “holds” a state, one of ordinary skill would understand that but for the action on the part of the device to preserve the state, the state would be subject to change. (This view of “holding” a state also is consistent with DeRoo’s usage of the term “held in reset” to describe how CPU 702 preempts SCP 706, as discussed above.)

40. It is my expert opinion that neither DeRoo’s address latch enable signal nor any other aspect of the operation of SCP 706 provides for “holding” a state in the manner just described. Specifically, as to DeRoo’s address latch enable signal, it is evident that this signal may be used to initiate access by SCP 706 to common memory device 704 during

shared mode operation (which is the only mode of operation in which DeRoo specifically describes the role of the address latch enable signal). However, DeRoo specifically states that the state of access that is initiated by this signal is unconditionally terminated by HUI 700 “after two rising and one falling edges of the SCP clock.” *See* DeRoo, col. 83, ll.55-57. That is, after SCP 706 asserts the address latch enable signal, DeRoo describes that CPU 702 automatically and unconditionally receives access after approximately two and a half clock cycles. *See id.* SCP 706 therefore cannot be said to preserve or “hold” the access state, because regardless of SCP 706’s activity, this state is terminated by HUI 700.

41. This is further demonstrated by inspection of DeRoo’s Verilog source code. In the source code, the signal flashSCP determines which of SCP 706 or CPU 702 has access to common memory device 704 at a given time. This signal is set by the falling edge of the address latch enable signal SCPALE. Further, the signal flashSCP is unconditionally cleared by the rising edge of a delayed version of itself (clr\_flashSCP). That is, the source code demonstrates that once set by SCP 706, flashSCP is reset by HUI 700. *See* DeRoo, cols. 129, 131.

42. Beyond the specific question of the address latch enable signal, it is my opinion that SCP 706 is not capable of holding any particular system state as the term “hold” would be understood by one of ordinary skill. As noted above in paragraphs 16-22, CPU 702 can arbitrarily preempt SCP 706 at any time, specifically by holding SCP 706 in reset. As noted above in paragraph 32, when a device is placed in a reset state as that term is used by DeRoo, the device is by definition unable to preserve its previous state, because the act of resetting causes the device to change its state to the defined reset state. Thus, because SCP 706 can be reset at any time by CPU 702, SCP 706 cannot “hold” a system state as recited in various claims of the ’576 patent.

43. Because “holding” a state and “releasing” a state are complementary concepts, it follows from the above points that SCP 706 also cannot be said to “release” a particular system state. With respect to the address latch enable signal, as noted above in paragraphs 40-41, although this signal is described by DeRoo as initiating SCP access

during shared mode, it is not responsible for releasing that state. Instead, as discussed above, the access state is unconditionally terminated by HUI 700 regardless of the state of SCP 706. Even when SCP 706 is operating in exclusive mode, DeRoo does not describe it as “releasing” or relinquishing this state when CPU 702 requests exclusive access. Rather, as discussed above, by resetting SCP 706, CPU 702 merely asserts its demand for exclusive access regardless of whether or not SCP 706 is prepared to release its own exclusive access.

44. More generally, because SCP 706 is not capable of “holding” a state as discussed above, there is logically nothing for SCP 706 to “release.” That is, it would be nonsensical to one of ordinary skill in the art to speak of a device “releasing” a state when it cannot “hold” the state to begin with.

45. Thus, it is my expert opinion that DeRoo fails to disclose that a microcontroller is configured to “hold the system in the first state” and “release the system from the first state” as recited by claims 1 and 6 of the ’576 patent.

*F. The Office Action’s proposed modification to DeRoo to allow SCP 706 to reset CPU 702 would change a fundamental principle of operation of DeRoo and would render DeRoo unsatisfactory for its intended purpose.*

46. Claim 11 recites a microcontroller that is configured to, “in response to a power on reset, hold a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory.” Claim 17 similarly recites “in response to a power on reset, holding a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory.” In rejecting claims 11 and 17, the Office Action acknowledges that DeRoo does not actually disclose numerous features of these claims, included the one just quoted. *See* Office Action at 9. However, the Office Action alleges that it would have been obvious to modify DeRoo such that SCP 706 could reset CPU 702, instead of merely placing CPU 702 into a “wait state.” *See id.* at 10.

47. As explained in greater detail below, it is my expert opinion that modifying DeRoo in the manner proposed by the Office Action would result in fundamentally

changing a core operating principle of DeRoo's design. Additionally, such a modification would increase the system's susceptibility to failure, for example by increasing the possibility of deadlocks that could hang the system, or of arbitrary resets of CPU 702 by SCP 706, either of which could significantly impair a user's ability to use DeRoo's system as a personal computer system, which is DeRoo's intended purpose. Accordingly, it is my expert opinion that given the implications of the Office Action's proposed modification to DeRoo, one of ordinary skill in the art would not find it obvious to make the proposed modification.

48. If a computing system includes more requesters to a shared device than the shared device can concurrently service, the possibility of conflicting requests for access to the shared device is endemic to the system. It is a fundamental principle of computing system design that if conflicts to a shared device could occur, some protocol, scheme, or algorithm must be included in the system design in order to resolve such conflicts. That is, to ensure proper system operation, the possibility of conflicts must be specifically and affirmatively accounted for in the system design; simply ignoring the possibility in the hopes that conflicts will not actually occur will almost certainly result in system malfunction.

49. Numerous design approaches exist with respect to the problem of conflict resolution, ranging from in sophistication from simple to complex, and exhibiting different properties with respect to how the resulting system resolves conflicts. For example, a system architect might choose to implement a conflict resolution design in which conflicts among devices are resolved through statically-assigned device priorities. In such a system, different devices may be assigned different levels of priority with respect to the shared device that are fixed. When a conflict between two such devices occurs, the higher-priority device "wins"—i.e., it receives its requested access, while the lower-priority device is forced to wait. A static or fixed-priority scheme is relatively simple to implement. However, because it effectively is a "winner-take-all" approach, such a scheme will not guarantee that lower-priority devices will have access to the shared resource in the event that higher-priority devices tend to monopolize it.

50. Other possible examples of design solutions for conflict resolution are possible. For example, a system architect might use a state machine that enforces a known order or pattern of device accesses, such that when conflicts occur, devices are allowed to proceed in an alternating or round-robin fashion, or according to some other pattern (e.g., the defined pattern may be designed to favor some devices over others by granting some devices more access opportunities than others). In other cases, the winner of a conflict may be chosen through random or pseudorandom techniques, or by taking into account historical usage patterns of various devices, e.g., in order to resolve conflicts in a manner that promotes a selected fair use policy. For example, selecting the winner of a conflict may take into account how relatively frequent and/or recent a particular device's historical access to the shared resource has been. Granting access to devices that have less recently or frequently used the shared resource may tend to prevent resource starvation.

51. Regardless of the particular design solution that is selected for conflict resolution, one of ordinary skill in the art would recognize that the chosen solution represents a fundamental decision regarding the architecture of the system—one that has potentially wide-ranging implications with respect to the various hardware and software components of the system. Specifically, whether the conflict resolution solution is based on static priority or dynamic arbitration, the various system components may be designed to specifically rely on the chosen solution.

52. Once a system has been designed for a particular conflict resolution strategy, the system's various components may not operate efficiently under a different strategy, and may simply not operate at all. For example, in a system using static device priority for conflict resolution, simply changing the priority of devices may result in drastic changes in the overall behavior of the system, including complete system failure.

53. As a general principle, altering the conflict resolution strategy around which a system is designed represents a fundamental change regarding a central operating principle of the system, and a change in the design assumptions on which various system components rely. At a minimum, substantial testing would be required to verify that the

system behaves as intended using the new conflict resolution strategy. In most cases, redesign of system components would be needed to properly reflect the changes to the system's operating principles engendered by the new conflict resolution strategy.

54. As an example of how system components may rely on assumptions concerning a system's conflict resolution strategy, consider a system that employs a static priority scheme in which a processor has priority over all other devices with respect to a shared device. Software executing on the processor, or other hardware elements associated with the processor, may specifically be designed with this priority scheme in mind, i.e., with the understanding that the processor's access to the shared device will never be delayed or impeded on account of another device in the system. Thus, for example, time-sensitive processes executing on the processor may be designed with the assumption that they will enjoy immediate access to the shared device whenever necessary.

55. However, if the conflict resolution strategy were changed such that the processor no longer had priority over all other devices—e.g., if, instead, some form of arbitration for the shared device were used—the assumption relied on by other system components no longer holds. Under the new strategy, components such as the time-sensitive processes mentioned above may encounter phenomena, such as delays, that would not have been possible under the original conflict resolution strategy. If a time-sensitive process incurs an unexpected delay (e.g., due to arbitration), it may miss the deadline that renders it time-sensitive. As a result, the process may malfunction. To prevent such an outcome, substantial redesign and testing would be required to ensure that the system operates as intended under the changed conflict resolution strategy.

56. Turning specifically to the details of DeRoo's system, as noted above in paragraph 22, DeRoo discloses a system in which, by virtue of the basic organization of the system, a processor's (i.e., CPU 702's) access priority with respect to the shared memory is unconditionally higher than the microcontroller's (i.e., SCP 706's). This corresponds to a fixed or static priority scheme of the sort discussed in paragraph 49 above. One of ordinary skill would recognize DeRoo's choice of a static priority scheme as a basic architectural feature of DeRoo's system organization that gives rise to a

fundamental principle of operation: when CPU 702 demands access to common memory device 704, CPU 702 will unconditionally receive such access. In order for DeRoo's system to operate correctly, the other components of the system need to rely on and reflect this principle of operation in their own behavior. For example, software that executes on CPU 702 may reasonably rely on having unfettered access to common memory device 704, whereas SCP 706 cannot rely on such access.

57. One of ordinary skill in the art would appreciate that if DeRoo were altered such that CPU 702 no longer was guaranteed higher priority than SCP 706 with respect to common memory device 704, such a modification would alter a fundamental principle of operation of DeRoo. That is, as described in detail in paragraphs 51-54 above, a system's conflict resolution scheme is central to the system's principle of operation; it in turn drives other design decisions with respect to the system and its components. It is highly likely that DeRoo's system would have to be redesigned in order to work properly if such a modification to DeRoo's conflict resolution scheme were made.

58. The modification proposed in the Office Action is exactly the sort of modification just referred to. If, as the Office Action proposes, DeRoo were modified such that SCP 706 could reset CPU 702 in the same manner that CPU 702 can reset SCP 706, the necessary result would be that CPU 706 would no longer be guaranteed higher priority than SCP 706 with respect to common memory device 704. That is, following such a modification, CPU 702 could no longer expect unfettered access to common memory device 704, but would rather be subject to substantial interference or delay owing to its being reset by SCP 706.

59. Therefore, it is my expert opinion that the modification of DeRoo proposed in the Office Action would change the principle of operation of DeRoo.

60. In addition to changing a core principle of operation of DeRoo, it is my expert opinion that the modification proposed in the Office Action would substantially degrade the performance of DeRoo, rendering DeRoo unsuitable for its intended purpose of use within a personal computer system.

61. First, providing SCP 706 with the ability to reset CPU 702 in a manner that is symmetrical to the ability of CPU 702 to reset SCP 706 would mean that both devices would have the ability to concurrently attempt to reset each other. If both CPU 702 and SCP 706 were to succeed in holding each other in reset, then DeRoo's system would effectively be deadlocked in reset with neither device able to proceed. The likely result of this scenario would be a system hang that would require, e.g., that the system be power cycled in order to restore normal system operation, which would likely result in loss of user data.

62. Second, DeRoo specifically describes that the intended purpose of the arrangement of SCP 706, CPU 702, and HUI 700 shown in Figure 20 is within a personal computer, where CPU 702 corresponds to the general purpose processor of the personal computer. *See, e.g.*, DeRoo, col. 81, ll.37-42; col. 4, ll.24-26. One of ordinary skill in the art would expect that in the context of a personal computer, CPU 702 would be expected to run an operating system (such as, e.g., a version of Microsoft Windows™) as well as various user applications, such as web browsers, word processors, and the like.

63. The Office Action's proposed modification to DeRoo would give SCP 706 the ability to arbitrarily reset CPU 702 in the same manner in which CPU 702 can reset SCP 706. As noted above in paragraph 32, to one of ordinary skill in the art, "resetting" a device would entail unconditionally placing the device in a consistent, reproducible reset state in such a manner that the device would not be capable of simply resuming an operation that was in progress at the time the device was reset. If SCP 706 had this reset ability with respect to CPU 702, it could arbitrarily erase the current operating state of CPU 702, interrupting any operating system or application software that was executing at the time. This would effectively amount to spontaneously rebooting the personal computer, and would almost assuredly result in data loss if implemented as the Office Action proposes.

64. Thus, the Office Action's proposed modification to DeRoo would increase the susceptibility of DeRoo's personal computer system both to system hangs and spontaneous resetting/rebooting of CPU 702, both highly undesirable failure modes that



are likely to result in user data loss. Such phenomena would significantly degrade the performance of DeRoo's personal computer system.

65. Therefore, it is my expert opinion that the modification of DeRoo proposed in the Office Action would render DeRoo unsatisfactory for its intended purpose of use within a personal computer system.

66. It is my understanding that if a proposed modification to a prior art reference would either change the principle of operation of the prior art reference or render the prior art reference unsatisfactory for its intended purpose, the proposed modification cannot support a prima facie case of obviousness of a claimed invention. Because in my expert opinion, the proposed modification to DeRoo would produce both of these impermissible results, claims 11 and 17 are not obvious in view of DeRoo.

67. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

4/19/2012

\_\_\_\_\_  
Date



\_\_\_\_\_  
Thomas M. Conte

# **EXHIBIT 1**

# THOMAS MARTIN CONTE

## CURRICULUM VITAE

### POSITIONS HELD

|   |                       |
|---|-----------------------|
| Professor, School of Electrical and Computer Engineering, College of Engineering, and<br>Professor, School of Computer Science, College of Computing (joint appointment)<br>Georgia Institute of Technology | 8/11 – <i>present</i> |
| Professor, School of Computer Science, College of Computing<br>Georgia Institute of Technology  | 7/08 – 8/11           |
| Professor, Department of Electrical and Computer Engineering<br>North Carolina State University   | 8/02 – 7/08           |
| Director, Center for Efficient, Scalable and Reliable Computing<br>North Carolina State University  | 1/07 – 7/08           |
| Director, Center for Embedded Systems Research<br>North Carolina State University   | 1/01 – 12/06          |
| Chief Microarchitect and Manager, Back End Compiler Development<br>BOPS, Inc. (VLIW DSP startup, while on leave from NCSU for AY00-01)  | 6/00 – 6/01           |
| Representative to EDN Embedded Benchmarking Consortium<br>BOPS, Inc.  | 6/00 – 6/01           |
| Associate Professor, Department of Electrical and Computer Engineering<br>North Carolina State University   | 7/98 – 7/02           |
| Assistant Professor, Department of Electrical and Computer Engineering<br>North Carolina State University   | 8/95 – 8/98           |
| Assistant Professor, Department of Electrical and Computer Engineering<br>University of South Carolina  | 8/92 – 8/95           |

### EDUCATION

Doctor of Philosophy, Electrical Engineering, University of Illinois, Urbana-Champaign: 1992  
Master of Science, Electrical Engineering, University of Illinois, Urbana-Champaign: 1988  
Bachelor of Electrical Engineering, University of Delaware: 1986

### HONORS

- IEEE Fellow (citation: “*for contributions to computer architecture, compiler code generation and performance evaluation*”)
- Young Alumni Achievement Award, Dept. of ECE, University of Illinois at Urbana-Champaign, 2004
- National Science Foundation Faculty Early Career Development (CAREER) Award, 1996
- IBM Partnership Award for Faculty Development, 1995, 1996
- Best paper awards: “*Dynamic rescheduling: A technique for object code compatibility in VLIW architectures,*” 1995 *International Symposium on Microarchitecture*; “*A technique to determine power-efficient, high-performance superscalar processors,*” 1995 *Hawaii International Conference on System Sciences*; “*Architectural resource requirements of contemporary benchmarks: A wish list,*” 1993 *Hawaii International Conference on System Sciences*
- Professional societies: *IEEE (fellow), ACM (member)*
- Honor societies: *Tau Beta Pi* (engineering), *Eta Kappa Nu* (electrical engineering)
- While a student: President of *University of Delaware IEEE Student Branch*, 1985; *University of Delaware Engineering Scholar*

## PUBLICATION HISTORY

### BOOKS

- [1] T. M. Conte and C. E. Gimarc, eds., *Fast Simulation of Computer Architectures*, Kluwer Academic Publishers: Boston, MA 7 1995, ISBN 0-7923-9593-X.

### BOOK CHAPTERS

- [1] T. M. Conte and K. N. P. Menezes, "The effects of traditional compiler optimizations on superscalar architectural design," in *The Interaction of Compilation Technology and Computer Architecture* (D. J. Lilja and P. L. Bird, eds.), Kluwer Academic Publishers: Boston, MA, 1994, pp. 119-136.
- [2] T. M. Conte and W. W. Hwu, "Advances in benchmarking techniques: New standards and quantitative metrics," in *Advances in Computers*, vol. 41, (M. V. Zelkowitz, ed.), Academic Press: New York, 1995.
- [3] T. M. Conte, "Superscalar and VLIW Processors," in *Handbook of Parallel and Distributed Computing*, (A.-Y. Zomaya, ed.), McGraw-Hill: New York, 1995.
- [4] T. M. Conte, "Superscalar and VLIW processors (instruction-level parallelism)", *Encyclopedia of Electrical and Electronics Engineering* (J. G. Webster, ed.), John Wiley & Sons: New York, NY, 1998.
- [5] T. M. Conte and P. D. Bryan, "Statistical sampling for processor and cache simulation," *Performance Evaluation and Benchmarking*, (L. John and L. Eeckhout, eds.), CRC Press, 2006.
- [6] T. M. Conte, "Appendix D: Embedded systems," in *Computer Architecture: A Quantitative Approach*, 4<sup>th</sup> ed. (J. Hennessy and D. Patterson), Morgan-Kaufmann: San Francisco, 2006.

### JOURNAL PAPERS

- [1] T. M. Conte and W. W. Hwu, "Benchmark characterization," *IEEE Computer*, pp. 48-56, Jan. 1991.
- [2] W. Y. Chen, P. P. Chang, T. M. Conte and W. W. Hwu, "The effect of code expanding optimizations on instruction cache design," *IEEE Transactions on Computers*, vol. C-42, no. 9, pp. 1045-1057, Sep. 1993.
- [3] W. W. Hwu and T. M. Conte, "The susceptibility of programs to context switching," *IEEE Transactions on Computers*, vol. C-43, no. 9, pp. 993-1003, Sep. 1994.
- [4] T. M. Conte, B. A. Patel, K. N. Menezes and J. S. Cox, "Hardware-based profiling: An effective technique for profile-driven optimization," *International Journal of Parallel Programming*, vol. 24, no. 2, Feb. 1996.
- [5] T. M. Conte and S. W. Sathaye, "Optimization of VLIW object code compatibility systems employing dynamic rescheduling," *International Journal of Parallel Programming*, vol. 25, no. 2, Feb. 1997.
- [6] T. M. Conte, P. K. Dubey, M. D. Jennings, R. B. Lee, S. Rathnam, M. Schlansker, P. Song, A. Wolfe, "Challenges to combining general-purpose and multimedia processors into one package," *IEEE Computer*, vol. 30, no. 12, Dec. 1997.
- [7] M. Schlansker, T. M. Conte, J. Dehnert, K. Ebcioglu, J. Fang, C. Thompson, "Compiling for Instruction-Level Parallelism," *IEEE Computer*, vol. 30, no. 12, Dec. 1997.
- [8] P. Bose and T. M. Conte, "Performance analysis and its impact on design", *IEEE Computer*, vol. 31, no. 5, May 1998.
- [9] T. M. Conte, M. A. Hirsch, and W. W. Hwu, "Combining trace sampling with single pass methods for efficient cache simulation," *IEEE Transactions on Computers*, vol. C-47, no. 6, Jun. 1998.
- [10] S. Banerjia, S. W. Sathaye, K. N. Menezes and T. M. Conte, "MPS: Miss path scheduling for multiple-issue processors," *IEEE Transactions on Computers*, vol. C-47, no. 12, Dec. 1998.
- [11] M. D. Jennings and T. M. Conte, "Subword extensions for video processing on mobile systems," *IEEE Concurrency*, July-September, 1998, pp. 13-16.
- [12] P. Bose, T. M. Conte, T. M. Austin, "Challenges in processor modeling and validation," *IEEE Micro*, May-June, 1999, pp. 2-6.

- [13] T. M. Conte, S. W. Sathaye, K. N. Menezes, and M. C. Toburen "System-level power consumption modeling and tradeoff analysis techniques for superscalar processor design," *IEEE Transactions on VLSI Systems*, vol. 8, no. 2, Apr. '00, pp.129-137.
- [14] T. M. Conte and S. W. Sathaye, "Properties of rescheduling size invariance for dynamic rescheduling-based VLIW cross-generation compatability", *IEEE Transactions on Computers*, vol. C-49, no. 8, Aug. '00, pp. 814-825.
- [15] T. M. Conte, "Choosing the brain(s) of an embedded system," *IEEE Computer*, vol. 35, no. 7., Jul. '02, pp. 106-107.
- [16] C. Fu, J. T. Bodine, T. M. Conte, "Modeling value speculation: An optimal edge selection problem," *IEEE Transactions on Computers*, vol. C-52, no. 3, Mar. '03, pp. 277-292.
- [17] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte. "Adaptive Mode Control: A Static-Power-Efficient Cache Design". *ACM Transactions in Embedded Computing Systems (TECS)*, vol. 2, no. 3, Aug. '03, pp. 347-372.
- [18] P. Mehrotra, V. Rao, T. M. Conte and P. D. Franzon, "Optimal Chip Package Co-design for High Performance DSP," *IEEE Transactions on Advanced Packaging*, vol. 28, no. 2, May '05, pp. 288 – 297.
- [19] A. Bechini, T. M. Conte, C. A. Prete, "Opportunities and challenges in embedded systems," *IEEE Micro*, July-August, '04, pp. 2-3.
- [20] H. Zhou and T. M. Conte, "Enhancing memory-level parallelism via recovery-free value prediction," *IEEE Transactions on Computers*, vol. C-54, no. 7, Jul. '05, pp. 897-912.
- [21] E. Ozer and T. M. Conte, "High-performance and low-cost dual-thread VLIW processor using WELD architecture paradigm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 12, Dec. '05.
- [22] S. Sharma, J. G. Beu and T. M. Conte, "Spectral prefetcher: An effective mechanism for L2 cache prefetching," *ACM Transactions on Architecture and Code Optimization*, vol. 2 , no. 4, Dec. '05, pp. 423-450.
- [23] J. A. Poovey, M. Levy, S. Gal-On, T. M. Conte, "A benchmark characterization of the EEMBC benchmark suite" *IEEE Micro*, Sep.-Oct. '09.

#### PEER-REVIEWED CONFERENCE PUBLICATIONS

- [1] W. W. Hwu and T. M. Conte, "A simulation study of simultaneous vector prefetch performance in multiprocessor memory subsystems (extended abstract)," in *Proceedings of the ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'89)*, (Berkeley, CA), p. 227, May 1989.
- [2] W. W. Hwu, T. M. Conte, and P. P. Chang, "Comparing software and hardware schemes for reducing the cost of branches," in *Proceedings of the 16<sup>th</sup> Annual International Symposium Computer Architecture (ISCA-16)*, (Jerusalem, Israel), pp. 224-233, June 1989.
- [3] T. M. Conte and W. W. Hwu, "Benchmark characterization for experimental system evaluation," in *Proceedings of the 23<sup>rd</sup> Hawaii International Conference on System Sciences*, Vol. 1, (Kona, HI), pp. 6-18, Jan. 1990.
- [4] T. M. Conte and W. W. Hwu, "Benchmark characterization," in *Proceedings of the 24<sup>th</sup> Hawaii International Conference on System Sciences*, vol. 1, (Kauai, HI), pp. 364-372, Jan. 1991.
- [5] T. M. Conte and W. W. Hwu, "Systematic prototyping of superscalar computer architectures," in *Proceedings of the 3<sup>rd</sup> IEEE International Workshop on Rapid System Prototyping*, (Research Triangle Park, NC), June 1992.
- [6] T. M. Conte, "Tradeoffs in processor/memory interfaces for superscalar processors," in *Proceedings of the 25<sup>th</sup> Annual International Symposium on Microarchitecture (MICRO-25)*, (Portland, OR), pp. 202-205, Dec. 1992.

- [7] T. M. Conte, "Architectural resource requirements of contemporary benchmarks: A wish list," in *Proceedings of the 26<sup>th</sup> Hawaii International Conference on System Sciences*, vol. 1, (Maui, HI), pp. 517-529, Jan. 1993. (winner: *best paper*)
- [8] T. M. Conte and W. Mangione-Smith, "Determining cost-effective multiple issue processor designs," in *Proceedings of the 1993 International Conference on Computer Design (ICCD'93)*, (Cambridge, MA), Oct. 1993.
- [9] T. M. Conte, B. A. Patel, and J. S. Cox, "Using branch handling hardware to support profile-driven optimization," in *Proceedings of the 27<sup>th</sup> Annual International Symposium on Microarchitecture (MICRO-27)*, (San Jose, CA), pp. 12-21, Dec. 1994.
- [10] T. M. Conte, K. N. P. Menezes and S. A. Sathaye, "A technique to determine power efficient, high-performance superscalar processors," in *Proceedings of the 28<sup>th</sup> Hawaii International Conference on System Sciences*, vol. 1, (Maui, HI), pp. 324-333, Jan. 1995. (winner: *best paper*)
- [11] J. S. Cox, D. P. Howell, and T. M. Conte, "Commercializing profile-driven optimization," in *Proceedings of the 28<sup>th</sup> Hawaii International Conference on System Sciences*, vol. 1, (Maui, HI), pp. 221-228, Jan. 1995.
- [12] T. M. Conte, K. N. Menezes, P. M. Mills and B. A. Patel, "Optimization of instruction fetch mechanisms for high issue rates," in *Proceedings of the 22<sup>nd</sup> Annual International Symposium on Computer Architecture (ISCA-22)*, (Santa Margherita, Italy), Jun. 1995.
- [13] A. Singla and T. M. Conte, "Bipartitioning for hybrid FPGA-software simulation," in *Proceedings of the 1996 IEEE International Conference on VLSI Design*, (Bangalore, India), Jan. 1996.
- [14] T. M. Conte and S. W. Sathaye, "Dynamic rescheduling: A technique for object code compatibility in VLIW architectures," in *Proceedings of the 28<sup>th</sup> Annual International Symposium on Microarchitecture (MICRO-28)*, (Ann Arbor, MI), Nov. 1995. (winner: *best paper*)
- [15] T. M. Conte, M. A. Hirsch, and K. N. Menezes, "Reducing state loss for effective trace sampling of superscalar processors," in *Proceedings of the 1996 International Conference, on Computer Design (ICCD'96)*, (Austin, TX), Oct. 1996.
- [16] T. M. Conte, S. Banerjia, S. Y. Larin, K. N. Menezes and S. W. Sathaye, "Instruction fetch mechanisms for VLIW architectures with compressed encodings," in *Proceedings of the 29<sup>th</sup> Annual International Symposium on Microarchitecture (MICRO-29)*, (Paris, France), Nov. 1996.
- [17] T. M. Conte, K. N. Menezes and M. A. Hirsch, "Accurate and practical profile-driven compilation using the profile buffer," in *Proceedings of the 29<sup>th</sup> Annual International Symposium on Microarchitecture (MICRO-29)*, (Paris, France), Nov. 1996.
- [18] T. M. Conte, S. Banerjia and S. W. Sathaye, "A persistent rescheduled-page cache for low overhead object code compatibility in VLIW architectures," in *Proceedings of the 29<sup>th</sup> Annual International Symposium on Microarchitecture (MICRO-29)*, (Paris, France), Nov. 1996.
- [19] S. Banerjia, W. A. Havanki, T. M. Conte, "Tree-region scheduling: A technique for extracting instruction level parallelism", in *Proceedings of the 1997 European conference in Parallel Processing*, (Passau, Germany), Aug., 1997.
- [20] K. N. Menezes, S. W. Sathaye and T. M. Conte, "Path prediction for high issue-rate processors," in *Proceedings of the 1997 International Conference on Parallel Architectures and Compilation Techniques (PACT'97)*, (San Francisco, CA), Nov. 1997.
- [21] W. A. Havanki, S. Banerjia and T. M. Conte, "Tree-region scheduling for wide-issue processors," in *Proceedings of the 1997 4<sup>th</sup> International Symposium on High-Performance Computer Architecture (HPCA-4)*, (Las Vegas), Feb. 1998.
- [22] C. Fu, M. D. Jennings, and T. M. Conte, "Value speculation scheduling for high performance processors," in *Proceedings of the 8<sup>th</sup> International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, (San Jose, CA), Oct., 1998.

- [23] E. Ozer, S. W. Sathaye, K. N. Menezes, S. Banerjia, M. D. Jennings and T. M. Conte, "A fast interrupt handling scheme for VLIW processors," in *Proceedings of the 1998 International Conference on Parallel Architectures and Compilation Techniques (PACT'98)*, (Paris, France), Oct. 1998.
- [24] E. Ozer, S. Banerjia, T. M. Conte, "Unified assign and schedule: A new approach to scheduling for clustered register file microarchitectures," in *Proceedings of the 31<sup>st</sup> Annual International Symposium on Microarchitecture*, (Dallas, TX), Nov. 1998.
- [25] S. Y. Larin and T. M. Conte, "Compiler-driven cached code compression schemes for embedded ILP processors," in *Proceedings of the 32<sup>nd</sup> Annual International Symposium on Microarchitecture (MICRO-32)*, (Haifa, Israel), Nov. 1999.
- [26] K. M. Hazelwood and T. M. Conte, "A lightweight algorithm for dynamic if-conversion during dynamic optimization," in *Proceedings of the 2000 International Conference on Parallel Architectures and Compilation Techniques (PACT'00)*, (Philadelphia, PA), Oct. 2000.
- [27] H. Zhou, M. C. Toburen, E. Rotenberg and T. M. Conte, "Adaptive mode control: A static-power-efficient cache," in *Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques (PACT'01)*, (Barcelona, Spain), Sep. 2001.
- [28] E. Ozer and T. M. Conte, "Weld: A multithreading technique towards latency-tolerant VLIW processors," in *Proceedings of the 8<sup>th</sup> International Conference on High Performance Computing (HiPC-8)*, (Hyderabad, India), Dec. 2001.
- [29] H. Zhou, M. D. Jennings, and T. M. Conte, "Tree traversal scheduling: A global scheduling technique for VLIW/EPIC processors," in *Proceedings of the 14<sup>th</sup> Annual Workshop on Languages and Compilers for Parallel Computing (LCPC'01)*, (Cumberland Falls, KY), August 2001.
- [30] H. Zhou and T.M. Conte, "Code size efficiency in global scheduling for ILP processors," in *Proceedings of the 6th Annual Workshop on the Interaction between Compilers and Computer Architectures (INTERACT-6)* held in conjunction with the 8th International Symposium on High Performance Computer Architecture (HPCA-8), (Cambridge, MA), February 2002.
- [31] G. G. Pechanek, S. Larin and T. Conte, "Any-size instruction abbreviation technique for embedded DSPs," in *Proceedings of the 15th Annual IEEE ASIC/SOC Conference*, (Rochester, NY), September 2002.
- [32] H. Zhou, J. Flanagan, T. M. Conte, "Detecting global stride locality in value streams," *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA-30)*, (San Diego, CA), June 2003, pp. 324–335.
- [33] H. Zhou and T. M. Conte, "Enhancing memory level parallelism via recovery-free value prediction," in *Proceedings of the 2003 International Conference on Supercomputing (ICS'03)* (San Francisco, CA), June 2003.
- [34] M. C. Rosier and T. M. Conte, "Treeregion Instruction Scheduling in GCC," in *Proceedings of the 2006 GCC Developers' Summit*, (Ottawa, Canada), June 2006.
- [35] P. D. Bryan, M. C. Rosier and T. M. Conte, "Reverse State Reconstruction for Sampled Microarchitectural Simulation," in *Proceedings of the 2007 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASSS'07)*, (San Jose, CA), April 2007.
- [36] P. D. Bryan and T. M. Conte, "Combining Cluster Sampling with Single Pass Methods for Efficient Sampling Regimen Design," in *Proceedings of the 2007 International Conference, on Computer Design (ICCD'07)*, (Lake Tahoe, CA), Oct. 2007.
- [37] B. V. Iyer and T. M. Conte, "A Power Model for Register-Sharing Structures," in *Proceedings of the 2008 IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES'08)*, (Milano, Italy), Sep. 2008.
- [38] B. V. Iyer, J. A. Poovey and T. M. Conte, "Energy-Aware Opcode Design," in *Proceedings of the 26th International Conference on Computer Design (ICCD'08)*, (Lake Tahoe, California), Oct. 12-15, 2008.

- [39] B. V. Iyer, J. G. Beu, and T. M. Conte, "Length Adaptive Processors: The solution for Energy/Performance Dilemma in Embedded Systems," *Workshop on Interaction Between Compilers and Computer Architecture (INTERACT-13), held in conjunction with HPCA-15*, (Raleigh, NC), Feb 16th, 2009.
- [40] B. V. Iyer and T. M. Conte, "On power and energy trends of IEEE 802.11n PHY," *Proceedings of the 12th International ACM Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2009)*, (Tenerife, Canary Islands, Spain), Oct. 26-19, 2009.
- [41] J. A. Poovey, B. P. Railing and T. M. Conte, "Parallel pattern detection for architectural improvements," *Proceedings of the 3rd USENIX Workshop on Hot Topics in Parallelism (HotPar'11)*, (Berkeley, CA), May 26–27, 2011.
- [42] R. Bheda, J. A. Poovey, J. G. Beu and T. M. Conte, "Energy Efficient Phase Change Memory Based Main Memory for Future High Performance Systems," *2<sup>nd</sup> International Green Computing Conference (IGCC'11)*, (Orlando, FL) July, 2011.
- [43] J. G. Beu, M. C. Rosier and T. M. Conte, "Manager-Client Pairing: A Framework for Implementing Coherence Hierarchies," *Proceedings of the 44<sup>th</sup> Annual International Symposium on Microarchitecture (MICRO-44)*, (Porto Alegre, Brazil), Dec., 2011.

#### PEER-REVIEWED TECHNICAL REPORTS AND NOTES

- [1] D. J. Farber, G. Delp, and T. M. Conte, "Thinwire protocol for connecting personal computers to the Internet," Request for Comment RFC-914, Defense Data Network Network Information Center (Internet NIC), 1984. [Internet protocol document]
- [2] T. M. Conte and W. W. Hwu, "A brief survey of benchmark usage in the architecture community," *Computer Architecture News*, vol. 19, no. 4, pp. 37-44, June 1991.

#### PATENTS

##### *Issued:*

- [1] US Patent No. 7,028,286, "Methods and apparatus for automated generation of abbreviated instruction set and configurable processor architecture," S. Y. Larin, P. G. Pechanek and T. M. Conte, issued April 11, 2006, 33 claims.
- [2] US Patent No. 7,865,692, "Methods and apparatus for automated generation of abbreviated instruction set and configurable processor architecture," S. Y. Larin, P. G. Pechanek and T. M. Conte, issued January 4, 2011, 6 claims.
- [3] US Patent No. 7,953,955, "Methods and apparatus for automated generation of abbreviated instruction set and configurable processor architecture," S. Y. Larin, P. G. Pechanek and T. M. Conte, issued May 31, 2011, 14 claims.
- [4] US Patent No. 8,131,970, "Compiler based cache allocation," T. M. Conte and A. Wolfe, issued March 6, 2012, 20 claims.

#### SERVICE

##### PROFESSIONAL SERVICE AT A NATIONAL OR INTERNATIONAL LEVEL:

- Societal: IEEE (Fellow); ACM (Member)
- Societal: IEEE (Fellow); ACM (Member)
- Editor in chief: *ACM Transactions on Architecture and Compiler Optimization*, 2009-present; *Journal of Instruction-Level Parallelism*, 1998–2000, 2003–2005
- Associate Editor: *IEEE Transactions on Computers*, 1999–2003, 2010–present; *ACM Transactions on Embedded Computer Systems*, 2001–2009; *ACM Transactions on Architecture and Compiler Optimization*, 2004–2009; *Journal of Instruction-Level Parallelism*, 2005–present; *IEEE Micro*, 2005–present; *IEEE Computer*, 2008–present



- Societal Chair: *IEEE Computer Society Technical Committee on Microarchitecture (TC-uARCH)*, 1998–2006; *ACM Special Interest Group on Microarchitecture (SIGMICRO)*, 2004–2006;
- Chair, *IEEE Computer Society Awards Committee*, 2008–present;
- Chair, *ACM/IEEE Eckert-Mauchly Award Committee*, 2004–2005;
- Chair, *IEEE Harry Goode Memorial Award Committee*, 2005–2008; *IEEE W. Wallace McDowell Award Committee*, 2005–2008
- Member, *IEEE Computer Society Board of Governors*, 2009–2011;
- Vice President of Publications, *IEEE Computer Society*, 2012–present;
- Member, *IEEE Computer Society Fellows Evaluation Committee*, 2005–2007
- Chair, *IEEE Computer Society Fellows Evaluation Committee*, 2011
- Member, *IEEE Fellows Evaluation Committee*, 2012–present
- Member, Steering Committees: *IEEE/ACM International Symposium on Microarchitecture*, 1998–present; *IEEE/ACM International Symposium on Code Generation and Optimization*, 2000–present
- General (Co-)Chair: 43<sup>rd</sup> *IEEE/ACM International Symposium on Microarchitecture*, 2010; 15<sup>th</sup> *IEEE International Symposium on High Performance Computer Architecture*, 2009; 39<sup>th</sup> *IEEE/ACM International Symposium on Microarchitecture*, 2006; *IEEE/ACM International Symposium on Code Generation and Optimization*, 2003
- Program Chair: *IEEE International Symposium on Workload Characterization*, 2009; *IEEE/ACM International Symposium on Code Generation and Optimization*, 2006; 7th *IEEE/ACM International Conference on Compilers, Architectures, and Synthesis for Embedded Systems*, 2005; 7th *IEEE/ACM International Conference on Compilers, Architectures, and Synthesis for Embedded Systems*, 2003; 30<sup>th</sup> *IEEE/ACM International Symposium on Microarchitecture*, 1997
- Program committees: *IEEE International Symposium on Performance Analysis of Systems and Software*, 2003–2006; *IEEE International Conference on Supercomputing*, 1997, 2006; *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 1993–1994, 1996–2005, 2007–2009, 2011–2012; *IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 1997, 2000, 2004, 2005, 2006, 2008, 2010–2012; *IEEE International Conference on High Performance Computer Architecture*, 1998, 2000–2008, 2010–2011; *IEEE International Conference on Parallel Architectures and Compilation Techniques*, 1998, 1999; *International Conference on Massively Parallel Computing Systems*, 1998; *IEEE International Computer Performance and Dependability Symposium*, 1998, 1999; *IEEE International Conference on Parallel Processing*, 1999; *ACM International Conference on Programming Language Design and Implementation*, 2003; *IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, 2003–2005
- Proposal reviewing: *National Science Foundation*, 1996, 1997, 2000, 2003, 2005, 2006

## PERSONAL

Married (Catherine Linder Conte), two children (ages 10 and 13).

# **EXHIBIT 2**

*ISA & EISA*

*By Edward Solari*

*Theory and Operation*

***Including PC, XT, AT, ISA, and EISA  
I/O Bus Operation***

**Annabooks  
San Diego**

**ISA & EISA THEORY AND OPERATION**  
by  
**Edward Solari**

PUBLISHED BY

Annabooks  
11848 Bernardo Plaza Ct., Suite 110  
San Diego, CA. 92128  
USA

619-673-0870

Copyright (C) Annabooks 1992, 1993

All Rights Reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the prior written permission of the publisher, except for the inclusion of brief quotations in a review.

Printed in the United States of America

ISBN 0-929392-15-9

Second Printing October 1993

Library of Congress Cataloging-in-Publication Data

Solari, Edward.

ISA & EISA : PC, XT, AT, and EISA PC I/O BUS timing / by Edward Solari.

p. cm.

Includes bibliographical references and index.

ISBN 0-929392-15-9

1. Microcomputers--Buses. 2. EISA (Computer bus) I. Title.

II. Title: ISA and EISA.

TK7895.B87S66 1992

621.39'81--dc20

92-39819  
CIP

The original IBM AT baseboard does not drive the IOCS16\* signal line because I/O platform resources were typically only accessed by the platform CPU. Consequently, some ISA add-on bus owner cards will execute 16 data bit access cycles without monitoring the IOCS16\* signal line. Also, for ISA compatible platforms, the IOCS16\* signal line was not qualified by the AENx signal line because no slot specific addressing was defined. For the repeated 256 byte block reflective of the platform I/O (see Chapter 2) the IOCS16\* signal line may be driven during all access cycles without problems.

On E-ISA and EISA compatible platforms, the repeated 256 byte blocks are used for slot specific I/O accesses; consequently, the IOCS16\* signal line must be qualified by the AENx signal line of each slot by all I/O add-on slave cards. Also, add-on bus owner cards must monitor the IOCS16\* signal line. See the ADD-ON BUS MASTER CARD section of Chapter 4 for more detailed information.

## IOCHRDY (8, 8/16 ISA or E-ISA & EISA CONNECTOR)

### ISA OR E-ISA PLATFORM

The I/O channel ready (IOCHRDY) signal line allows resources to indicate to the ISA or E-ISA bus master that additional cycle time is required. If the IOCHRDY signal line is not driven inactive, the cycle is either a no-wait-state or a standard cycle. If the IOCHRDY signal line is driven inactive, the cycle length is extended until the line is driven active again. An inactive IOCHRDY signal line causes an active SRDY\* signal to be ignored by the bus master.

On an ISA or E-ISA compatible platform, the IOCHRDY signal line must be driven inactive within a specific time of the COMMAND signal lines becoming active. When the IOCHRDY signal line is active, the cycle ends within a specific amount of time.

Both of these events occur asynchronously to the BCLK signal line on an ISA compatible platform. On an E-ISA platform, the IOCHRDY signal line is sampled relative to the BCLK signal line in addition to supporting the previously outlined "asynchronous" support. See the IOCHRDY SIGNAL LINE ENHANCEMENT section in this chapter.

When the bus owner is either the platform CPU, an ISA or E-ISA add-on bus owner card, or the refresh controller, the increase in cycle length is measured as "wait states". The resolution of the wait states is one BCLK signal line period. When the bus owner is the DMA controller, the resolution of the wait states is two BCLK signal line periods for DMA COMPATIBLE transfer cycles. The DMA controller on an E-ISA compatible platform allows each DMA channel to be programmed for different DMA transfer cycles. Each of these cycles has a different wait state resolution. See the DMA TRANSFER CYCLE section in Chapter 6 for more detailed information.

For a DMA transfer cycle, only the memory resource can inactivate the IOCHRDY signal line; the I/O resource must be able to accept the data within the length of a default cycle.

### *EISA PLATFORMS*

When an ISA, E-ISA, or EISA bus master is accessing an ISA or E-ISA resource, the IOCHRDY signal line is defined the same as for the ISA or E-ISA compatible platform. For an E-MIX access cycle, this signal line is sampled by the platform circuitry to determine the cycle length. For an I-MIX access cycle, this signal line is driven by the platform circuitry as a translation of the EXRDY signal line. This signal line is not defined when an EISA bus master is accessing an EISA resource.

On an EISA compatible platform, the IOCHRDY signal line is sampled relative to the BCLK signal line, in addition to supporting the previously outlined "asynchronous support". See the IOCHRDY SIGNAL LINE ENHANCEMENT section in this chapter.

The DMA controller on an EISA compatible platform allows each DMA channel to be programmed for different DMA transfer cycle types. Each of these cycles has a different wait state resolution. See the DMA TRANSFER CYCLE section in Chapter 6 for more detailed information.

For a DMA transfer cycle, only the memory participant can drive the IOCHRDY signal line. The I/O resource must be able to accept the data within the length of a default cycle.

### *ALL PLATFORMS*

The IOCHRDY signal line should not remain inactive during access or transfer cycles longer than 15.6 microseconds. Otherwise, a bus refresh cycle may not occur at the appropriate time and DRAM data will be lost. The IBM Technical Reference Manual specifies 2.5 microseconds for unknown reasons. One possible explanation might be to allow other bus masters to arbitrate and execute cycles prior to a another refresh occurring.

*The original IBM AT baseboard does not drive the IOCHRDY signal line. Consequently, ISA add-on bus owner cards will execute access cycles without monitoring the IOCHRDY signal line. On E-ISA and EISA platforms, the IOCHRDY signal line cannot be ignored. In order to properly access the slot specific I/O space and EISA compatible resources, all ISA or E-ISA add-on bus owner cards must monitor the IOCHRDY signal line. See the ADD-ON BUS MASTER CARD section of Chapter 4 for more detailed information.*

### *IOCHRDY SIGNAL LINE ENHANCEMENT*

As previously described, the IOCHRDY signal line must be inactive within a "specific" time after the COMMAND signal line becomes active for a ready cycle. The ready cycle is completed some time after 125 nanoseconds (120 nanoseconds for E-ISA and EISA compatible platforms) after the IOCHRDY signal line has

been driven active. This timing of the IOCHRDY signal line is fairly imprecise and tends to lower performance.

In order to enhance performance, the EISA bus specification has defined a SAMPLING WINDOW that relates the IOCHRDY and BCLK signal lines. Figure 5-2 outlines this SAMPLING WINDOW as viewed by the accessed memory or I/O resource and the transfer cycle resource. The SAMPLING WINDOW can be viewed in two parts: inactive and active.

For CASES A & B, the inactive part of the INITIAL SAMPLING WINDOW is between the rising and falling edges of the BCLK signal line. An IOCHRDY signal line inactive pulse of 31 nanoseconds (for a 20 nanoseconds minimum inactive pulse at bus master) within the inactive part of the window will cause a ready cycle to be executed. For CASE C, the minimum setup time of 21 nanoseconds to the falling edge of the BCLK signal line requires the receiving platform CPU or add-on bus owner card to support an actual setup time of 10 nanoseconds. The additional 11 nanosecond loss reflects allowances for bus settling and clock tolerances.

In the active part of the INITIAL SAMPLING WINDOW, the IOCHRDY signal line is sampled at the rising clock edge at the end of the INITIAL SAMPLING WINDOW. If the IOCHRDY signal line is sampled active, the ready cycle will be completed at the end of the next BCLK period. The IOCHRDY signal line is specified as an open collector device with a 1K pull-up resistor. For an E-ISA or EISA compatible platform with an 8 MHz BCLK signal line, assume a 10 nanosecond setup time at the platform CPU or add-on bus owner card, an 11 nanosecond signal line settling time, and a 104 nanosecond rise time, then the total time is 125 nanoseconds. A minimum INITIAL SAMPLING WINDOW of 125 nanoseconds and 20 nanosecond minimum pulse width results in a one wait state ready cycle not being guaranteed. For an



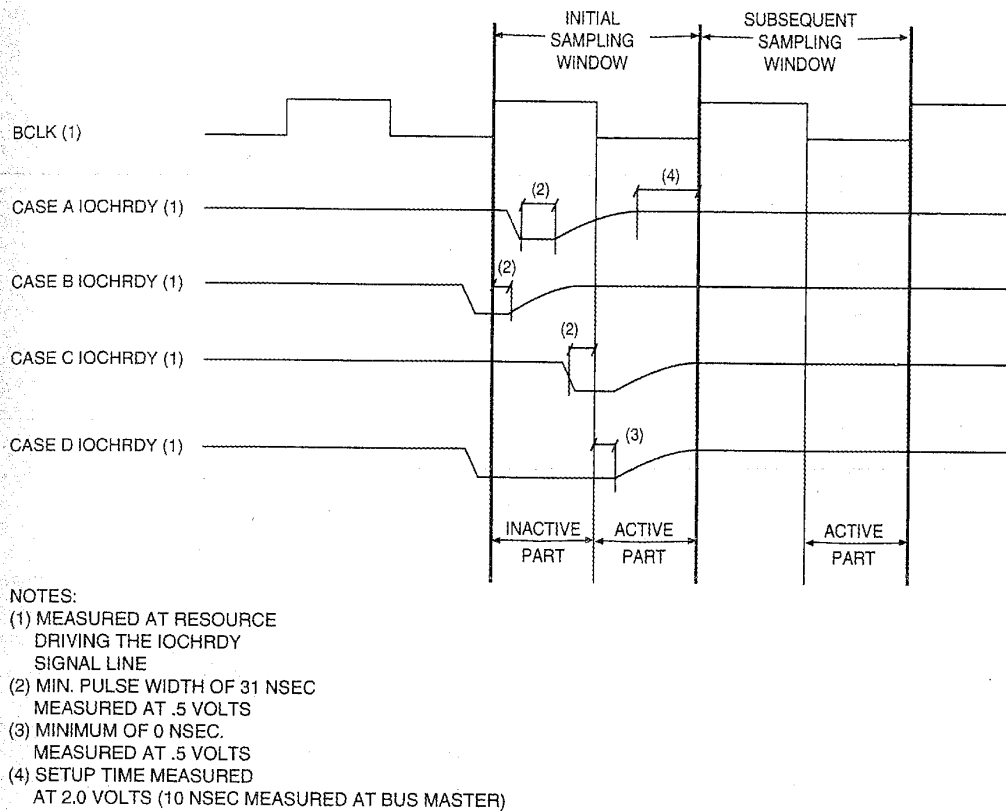


FIGURE 5-2: E-ISA AND EISA IOCHRDY REFERENCE TO THE BCLK SIGNAL LINE

E-ISA or EISA compatible platform with an 8.33 MHz BCLK signal line, the INITIAL SAMPLING WINDOW is 120 nanoseconds, the minimum inactive IOCHRDY signal line pulse width is 20 nanoseconds, and the rise time is now 104 nanoseconds. Consequently, in a fully loaded E-ISA or EISA platform, requirements for a guaranteed one wait state ready cycle are "just missed". To guarantee "conservatively" a one wait state ready cycle, the IOCHRDY signal line must be pulsed high, or a 300 ohm pull-up resistor must be used on the bus in that the rise time is only 29 nanoseconds instead of 104 nanoseconds.

If the IOCHRDY signal line is not sampled active at the end of the INITIAL SAMPLING WINDOW, SUBSEQUENT SAMPLING WINDOWS occur. These SUBSEQUENT SAMPLING WINDOWS only have an active part. The IOCHRDY signal line is sampled at the rising clock edge at the end of each sampling window. If the IOCHRDY signal line is sampled active, the ready cycle will be completed at the end of the next BCLK period. In order to insure that the IOCHRDY is sampled active, the following must be considered:

For CASE D, in order to insure only one wait state, the rise time due to a 1K resistor (for a fully loaded E-ISA or EISA backplane) of 104 nanoseconds is too long. The rise time due to a 300 ohm resistor has a rise time of 29 nanoseconds and will allow all the requirements to be met.

# **EXHIBIT 3**



## APPLICATION NOTE

AP-69

May 1980

# An Introduction to the Intel® MCS-51™ Single-Chip Microcomputer Family

John Wharton  
Microcontroller Applications

\*Intel Corporation 1980

AFN-01502A-01

1

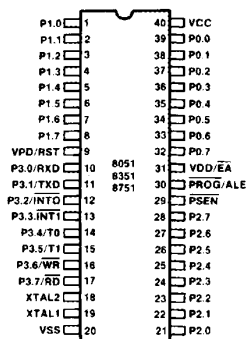


Figure 1a. 8051 Microcomputer Pinout Diagram

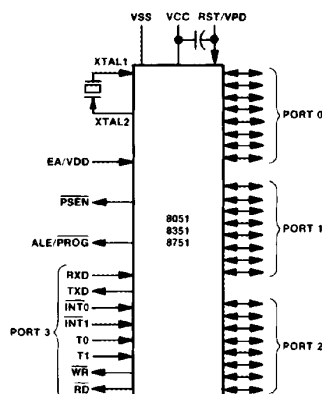


Figure 1b. 8051 Microcomputer Logic Symbol

## 1. INTRODUCTION

In 1976 Intel introduced the MCS-48™ family, consisting of the 8048, 8748, and 8035 microcomputers. These parts marked the first time a complete microcomputer system, including an eight-bit CPU, 1024 8-bit words of ROM or EPROM program memory, 64 words of data memory, I/O ports and an eight-bit timer counter could be integrated onto a single silicon chip. Depending only on the program memory contents, one chip could control a limitless variety of products, ranging from appliances or automobile engines to text or data processing equipment. Follow-on products stretched the MCS-48™ architecture in several directions: the 8049 and 8039 doubled the amount of on-chip memory and ran 83% faster; the 8021 reduced costs by executing a subset of the 8048 instructions with a somewhat slower clock; and the 8022 put a unique two-channel 8-bit analog-to-digital converter on the same NMOS chip as the computer, letting the chip interface directly with analog transducers.

Now three new high-performance single-chip microcomputers—the Intel® 8051, 8751, and 8031—extend the advantages of Integrated Electronics to whole new product areas. Thanks to Intel's new HMOS technology, the MCS-51™ family provides four times the program memory and twice the data memory as the 8048 on a single chip. New I/O and peripheral capabilities both increase the range of applicability and reduce total system cost. Depending on the use, processing throughput increases by two and one-half to ten times.

This Application Note is intended to introduce the reader to the MCS-51™ architecture and features. While it does not assume intimacy with the MCS-48™ product line on the part of the reader, he/she should be familiar with

some microprocessor (preferably Intel's, of course) or have a background in computer programming and digital logic.

## Family Overview

Pinout diagrams for the 8051, 8751, and 8031 are shown in Figure 1. The devices include the following features:

- Single-supply 5 volt operation using HMOS technology.
- 4096 bytes program memory on-chip (not on 8031).
- 128 bytes data memory on-chip.
- Four register banks.
- 128 User-defined software flags.
- 64 Kilobytes each program and external RAM addressability.
- One microsecond instruction cycle with 12 MHz crystal.
- 32 bidirectional I/O lines organized as four 8-bit ports (16 lines on 8031).
- Multiple mode, high-speed programmable Serial Port.
- Two multiple mode, 16-bit Timer/Counters.
- Two-level prioritized interrupt structure.
- Full depth stack for subroutine return linkage and data storage.
- Augmented MCS-48™ instruction set.
- Direct Byte and Bit addressability.
- Binary or Decimal arithmetic.
- Signed-overflow detection and parity computation.
- Hardware Multiple and Divide in 4  $\mu$ sec.
- Integrated Boolean Processor for control applications.
- Upwardly compatible with existing 8048 software.

AFN-01502A-04

All three devices come in a standard 40-pin Dual In-Line Package, with the same pin-out, the same timing, and the same electrical characteristics. The primary difference between the three is the on-chip program memory—different types are offered to satisfy differing user requirements.

The 8751 provides 4K bytes of ultraviolet-Erasable, Programmable Read Only Memory (EPROM) for program development, prototyping, and limited production runs. (By convention, 1K means  $2^{10} = 1024$ , 1k—with a lower case “k”—equals  $10^3 = 1000$ .) This part may be individually programmed for a specific application using Intel's Universal PROM Programmer (UPP). If software bugs are detected or design specifications change the same part may be “erased” in a matter of minutes by exposure to ultraviolet light and reprogrammed with the modified code. This cycle may be repeated indefinitely during the design and development phase.

The final version of the software must be programmed into a large number of production parts. The 8051 has 4K bytes of ROM which are mask-programmed with the customer's order when the chip is built. This part is considerably less expensive, but cannot be erased or altered after fabrication.

The 8031 does not have any program memory on-chip, but may be used with up to 64K bytes of external standard or multiplexed ROMs, PROMs, or EPROMs. The 8031 fits well in applications requiring significantly larger or smaller amounts of memory than the 4K bytes provided by its two siblings.

(The 8051 and 8751 automatically access external program memory for all addresses greater than the 4096 bytes on-chip. The External Access input is an override for all internal program memory—the 8051 and 8751 will each emulate an 8031 when pin 31 is low.)

Throughout this Note, “8051” is used as a generic term. Unless specifically stated otherwise, the point applies equally to all three components. Table 1 summarizes the quantitative differences between the members of the MCS-48™ and MCS-51™ families.

The remainder of this Note discusses the various MCS-51™ features and how they can be used. Software and/or hard-

ware application examples illustrate many of the concepts. Several isolated tasks (rather than one complete system design example) are presented in the hope that some of them will apply to the reader's experiences or needs.

A document this short cannot detail all of a computer system's capabilities. By no means will all the 8051 instructions be demonstrated; the intent is to stress new or unique MCS-51™ operations and instructions generally used in conjunction with each other. For additional hardware information refer to the Intel MCS-51™ Family User's Manual, publication number 121517. The assembly language and use of ASM51, the MCS-51™ assembler, are further described in the MCS-51™ Macro Assembler User's Guide, publication number 9800937.

The next section reviews some of the basic concepts of microcomputer design and use. Readers familiar with the 8048 may wish to skim through this section or skip directly to the next, “ARCHITECTURE AND ORGANIZATION.”

### Microcomputer Background Concepts

Most digital computers use the binary (base 2) number system internally. All variables, constants, alphanumeric characters, program statements, etc., are represented by groups of binary digits (“bits”), each of which has the value 0 or 1. Computers are classified by how many bits they can move or process at a time.

The MCS-51™ microcomputers contain an eight-bit central processing unit (CPU). Most operations process variables eight bits wide. All internal RAM and ROM, and virtually all other registers are also eight bits wide. An eight-bit (“byte”) variable (shown in Figure 2) may assume one of  $2^8 = 256$  distinct values, which usually represent integers between 0 and 255. Other types of numbers, instructions, and so forth are represented by one or more bytes using certain conventions.

For example, to represent positive and negative values, the most significant bit (D7) indicates the sign of the other seven bits—0 if positive, 1 if negative—allowing integer variables between -128 and +127. For integers with extremely large magnitudes, several bytes are manipulated together as “multiple precision” signed or unsigned integers—16, 24, or more bits wide.

Table 1. Features of Intel's Single-Chip Microcomputers

| EPROM Program Memory | ROM Program Memory | External Program Memory | Program Memory (Int/Max) | Data Memory (Bytes) | Instr. Cycle Time | Input/Output Pins | Interrupt Sources | Reg. Banks |
|----------------------|--------------------|-------------------------|--------------------------|---------------------|-------------------|-------------------|-------------------|------------|
| ...                  | 8021               | ...                     | 1K 1K                    | 64                  | 8.4 $\mu$ Sec     | 21                | 0                 | 1          |
| ...                  | 8022               | ...                     | 2K 2K                    | 64                  | 8.4 $\mu$ Sec     | 28                | 2                 | 1          |
| 8748                 | 8048               | 8035                    | 1K 4K                    | 64                  | 2.5 $\mu$ Sec     | 27                | 2                 | 2          |
|                      | 8049               | 8039                    | 2K 4K                    | 128                 | 1.36 $\mu$ Sec    | 27                | 2                 | 2          |
| 8751                 | 8051               | 8031                    | 4K 64K                   | 128                 | 1.0 $\mu$ Sec     | 32                | 5                 | 4          |

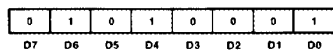
AFN-01502A-05

The letters "MCS" have traditionally indicated a system or family of compatible Intel® microcomputer components, including CPUs, memories, clock generators, I/O expanders, and so forth. The numerical suffix indicates the microprocessor or microcomputer which serves as the cornerstone of the family. Microcomputers in the MCS-48™ family currently include the 8048-series (8035, 8048, & 8748), the 8049-series (8039 & 8049), and the 8021 and 8022; the family also includes the 8243, an I/O expander compatible with each of the microcomputers. Each computer's CPU is derived from the 8048, with essentially the same architecture, addressing modes, and instruction set, and a single assembler (ASM48) serves each.

The first members of the MCS-51™ family are the 8051, 8751, and 8031. The architecture of the 8051-series, while derived from the 8048, is not strictly compatible; there are more addressing modes, more instructions, larger address spaces, and a few other hardware differences. In this Application Note the letters "MCS-51" are used when referring to *architectural* features of the 8051-series—features which would be included on possible future microcomputers based on the 8051 CPU. Such products could have different amounts of memory (as in the 8048/8049) or different peripheral functions (as in the 8021 and 8022) while leaving the CPU and instruction set intact. ASM51 is the assembler used by all microcomputers in the 8051 family.

Two digit decimal numbers may be "packed" in an eight-bit value, using four bits for the binary code of each digit. This is called Binary-Coded Decimal (BCD) representation, and is often used internally in programs which interact heavily with human beings.

Alphanumeric characters (letters, numbers, punctuation marks, etc.) are often represented using the American Standard Code for Information Interchange (ASCII) convention. Each character is associated with a unique seven-bit binary number. Thus one byte may represent



**Figure 2. Representation of Bits Within an Eight-Bit "Byte" (Value shown = 01010001 Binary = 81 decimal).**

a single character, and a word or sequence of letters may be represented by a series (or "string") of bytes. Since the ASCII code only uses 128 characters, the most significant bit of the byte is not needed to distinguish between characters. Often D7 is set to 0 for all characters. In some coding schemes, D7 is used to indicate the "parity" of the other seven bits—set or cleared as necessary to ensure that the total number of "1" bits in the eight-bit code is even ("even parity") or odd ("odd parity"). The 8051 includes hardware to compute parity when it is needed.

A computer program consists of an ordered sequence of specific, simple steps to be executed by the CPU one-at-a-time. The method or sequence of steps used collectively to solve the user's application is called an "algorithm."

The program is stored inside the computer as a sequence of binary numbers, where each number corresponds to one of the basic operations ("opcodes") which the CPU is capable of executing. In the 8051, each program memory location is one byte. A complete instruction consists of a sequence of one or more bytes, where the first defines the operation to be executed and additional bytes (if needed) hold additional information, such as data values or variable addresses. No instruction is longer than three bytes.

The way in which binary opcodes and modifier bytes are assigned to the CPU's operations is called the computer's "machine language." Writing a program directly in machine language is time-consuming and tedious. Human beings think in words and concepts rather than encoded numbers, so each CPU operation and resource is given a name and standard abbreviation ("mnemonic"). Programs are more easily discussed using these standard mnemonics, or "assembly language," and may be typed into an Intel® Intellec® 800 or Series II® microcomputer development system in this form. The development system can mechanically translate the program from assembly language "source" form to machine language "object" code using a program called an "assembler." The MCS-51™ assembler is called ASM51.

There are several important differences between a computer's machine language and the assembly language used as a tool to represent it. The machine language or instruction set is the set of operations which the CPU can perform while a program is executing ("at run-time"), and is strictly determined by the microcomputer hardware design.

The assembly language is a standard (though more-or-less arbitrary) set of symbols including the instruction set mnemonics, but with additional features which further simplify the program design process. For example, ASM51 has controls for creating and formatting a program listing, and a number of directives for allocating variable storage and inserting arbitrary bytes of data into the object code for creating tables of constants.

AFN-01502A-06

In addition, ASM51 can perform sophisticated mathematical operations, computing addresses or evaluating arithmetic expressions to relieve the programmer from this drudgery. However, these calculations can only use information known at "assembly time."

For example, the 8051 performs arithmetic calculations at run-time, eight bits at a time. ASM51 can do similar operations 16 bits at a time. The 8051 can only do one simple step per instruction, while ASM51 can perform complex calculations in each line of source code. However, the operations performed by the assembler may only use parameter values fixed at assembly-time, not variables whose values are unknown until program execution begins.

For example, when the assembly language source line,

```
ADD A,#(LOOP_COUNT + 1) * 3
```

is *assembled*, ASM51 will find the value of the previously-defined constant "LOOP\_COUNT" in an internal symbol table, increment the value, multiply the sum by three, and (assuming it is between -256 and 255 inclusive) truncate the product to eight bits. When this instruction is *executed*, the 8051 ALU will just add that resulting constant to the accumulator.

Some similar differences exist to distinguish number system ("radix") specifications. The 8051 does all computations in binary (though there are provisions for then converting the result to decimal form). In the course of writing a program, though, it may be more convenient to specify constants using some other radix, such as base 10. On other occasions, it is desirable to specify the ASCII code for some character or string of characters without referring to tables. ASM51 allows several representations for constants, which are converted to binary as each instruction is assembled.

For example, binary numbers are represented in the

assembly language by a series of ones and zeros (naturally), followed by the letter "B" (for Binary); octal numbers as a series of octal digits (0-7) followed by the letter "O" (for Octal) or "Q" (which doesn't stand for anything, but *looks* sort of like an "O" and is less likely to be confused with a zero).

Hexadecimal numbers are represented by a series of hexadecimal digits (0-9,A-F), followed by (you guessed it) the letter "H." A "hex" number must begin with a decimal digit; otherwise it would look like a user-defined symbol (to be discussed later). A "dummy" leading zero may be inserted before the first digit to meet this constraint. The character string "BACH" could be a legal label for a Baroque music synthesis routine; the string "0BACH" is the hexadecimal constant BACH<sub>16</sub>. This is a case where adding 0 makes a big difference.

Decimal numbers are represented by a sequence of decimal digits, optionally followed by a "D." If a number has no suffix, it is assumed to be decimal—so it had better not contain any non-decimal digits. "0BAC" is not a legal representation for anything.

When an ASCII code is needed in a program, enclose the desired character between two apostrophes (as in '#') and the assembler will convert it to the appropriate code (in this case 23H). A string of characters between apostrophes is translated into a series of constants; 'BACH' becomes 42H, 41H, 43H, 48H.

These same conventions are used throughout the associated Intel documentation. Table 2 illustrates some of the different number formats.

## 2. ARCHITECTURE AND ORGANIZATION

Figure 3 blocks out the MCS-51™ internal organization. Each microcomputer combines a Central Processing Unit, two kinds of memory (data RAM plus program ROM or EPROM), Input/Output ports, and the mode,

Table 2. Notations Used to Represent Numbers

| Bit Pattern     | Binary    | Octal | Hexa-Decimal | Decimal | Signed Decimal |
|-----------------|-----------|-------|--------------|---------|----------------|
| 0 0 0 0 0 0 0 0 | 0B        | 0Q    | 00H          | 0       | 0              |
| 0 0 0 0 0 0 0 1 | 1B        | 1Q    | 01H          | 1       | +1             |
| .....           | ..        | ..Q   | ..H          | ..      | ...            |
| 0 0 0 0 0 1 1 1 | 111B      | 7Q    | 07H          | 7       | +7             |
| 0 0 0 0 1 0 0 0 | 1000B     | 10Q   | 08H          | 8       | +8             |
| 0 0 0 0 1 0 0 1 | 1001B     | 11Q   | 09H          | 9       | +9             |
| 0 0 0 0 1 0 1 0 | 1010B     | 12Q   | 0AH          | 10      | +10            |
| .....           | ..        | ..Q   | ..H          | ..      | ...            |
| 0 0 0 0 1 1 1 1 | 1111B     | 17Q   | 0FH          | 15      | +15            |
| 0 0 0 1 0 0 0 0 | 10000B    | 20Q   | 10H          | 16      | +16            |
| .....           | ..        | ..Q   | ..H          | ..      | ...            |
| 0 1 1 1 1 1 1 1 | 111111B   | 177Q  | 7FH          | 127     | +127           |
| 1 0 0 0 0 0 0 0 | 10000000B | 200Q  | 80H          | 128     | -128           |
| 1 0 0 0 0 0 0 1 | 10000001B | 201Q  | 81H          | 129     | -127           |
| .....           | ..        | ..Q   | ..H          | ..      | ...            |
| 1 1 1 1 1 1 1 0 | 11111110B | 376Q  | 0FEH         | 254     | -2             |
| 1 1 1 1 1 1 1 1 | 11111111B | 377Q  | 0FFH         | 255     | -1             |

AFN-01502A-07



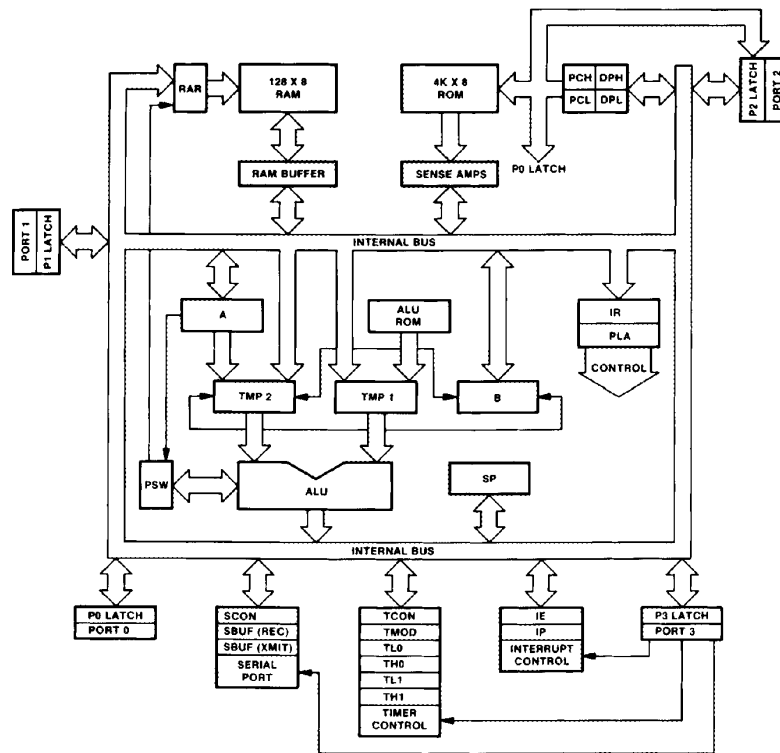


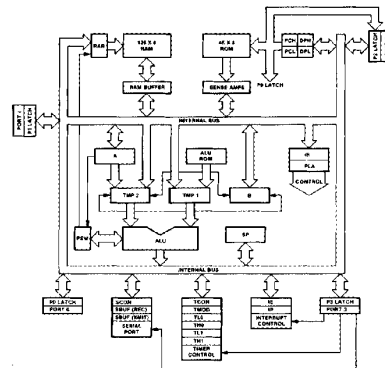
Figure 3. Block Diagram of 8051 Internal Structure

status, and data registers and random logic needed for a variety of peripheral functions. These elements communicate through an eight-bit data bus which runs throughout the chip, somewhat akin to indoor plumbing. This bus is buffered to the outside world through an I/O port when memory or I/O expansion is desired.

Let's summarize what each block does; later chapters dig into the CPU's instruction set and the peripheral registers in much greater detail.

#### Central Processing Unit

The CPU is the "brains" of the microcomputer, reading the user's program and executing the instructions stored therein. Its primary elements are an eight-bit Arithmetic/Logic Unit with associated registers A, B, PSW, and SP, and the sixteen-bit Program Counter and "Data Pointer" registers.



AFN-01502A-08

### Arithmetic Logic Unit

The ALU can perform (as the name implies) arithmetic and logic functions on eight-bit variables. The former include basic addition, subtraction, multiplication, and division; the latter include the logical operations AND, OR, and Exclusive-OR, as well as rotate, clear, complement, and so forth. The ALU also makes conditional branching decisions, and provides data paths and temporary registers used for data transfers within the system. Other instructions are built up from these primitive functions: the addition capability can increment registers or automatically compute program destination addresses; subtraction is also used in decrementing or comparing the magnitude of two variables.

These primitive operations are automatically cascaded and combined with dedicated logic to build complex instructions such as incrementing a sixteen-bit register pair. To execute one form of the compare instruction, for example, the 8051 increments the program counter three times, reads three bytes of program memory, computes a register address with logical operations, reads internal data memory twice, makes an arithmetic comparison of two variables, computes a sixteen-bit destination address, and decides whether or not to make a branch—all in two microseconds!

An important and unique feature of the MCS-51 architecture is that the ALU can also manipulate one-bit as well as eight-bit data types. Individual bits may be set, cleared, or complemented, moved, tested, and used in logic computations. While support for a more primitive data type may initially seem a step backwards in an era of increasing word length, it makes the 8051 especially well suited for controller-type applications. Such algorithms *inherently* involve Boolean (true/false) input and output variables, which were heretofore difficult to implement with standard microprocessors. These features are collectively referred to as the MCS-51™ “Boolean Processor,” and are described in the so-named chapter to come.

Thanks to this powerful ALU, the 8051 instruction set fares well at both real-time control and data intensive algorithms. A total of 51 separate operations move and manipulate three data types: Boolean (1-bit), byte (8-bit), and address (16-bit). All told, there are eleven addressing modes—seven for data, four for program sequence control (though only eight are used by more than just a few specialized instructions). Most operations allow several addressing modes, bringing the total number of instructions (operation/addressing mode combinations) to 111, encompassing 255 of the 256 possible eight-bit instruction opcodes.

### Instruction Set Overview

Table 4 lists these 111 instructions classified into five groups:

- Arithmetic Operations
- Logical Operations for Byte Variables
- Data Transfer Instructions
- Boolean Variable Manipulation
- Program Branching and Machine Control

MCS-48™ programmers perusing Table 4 will notice the absence of special categories for Input/Output, Timer/Counter, or Control instructions. These functions are all still provided (and indeed many new functions are added), but as special cases of more generalized operations in other categories. To explicitly list all the useful instructions involving I/O and peripheral registers would require a table approximately four times as long.

Observant readers will also notice that all of the 8048's page-oriented instructions (conditional jumps, JMPP, MOVP, MOVP3) have been replaced with corresponding but non-paged instructions. The 8051 instruction set is entirely *non-page-oriented*. The MCS-48™ “MOVP” instruction replacement and all conditional jump instructions operate relative to the program counter, with the actual jump address computed by the CPU during instruction execution. The “MOVP3” and “JMPP” replacements are now made relative to another sixteen-bit register, which allows the effective destination to be anywhere in the program memory space, regardless of where the instruction itself is located. There are even three-byte jump and call instructions allowing the destination to be *anywhere* in the 64K program address space.

The instruction set is designed to make programs efficient both in terms of code size and execution speed. No instruction requires more than three bytes of program memory, with the majority requiring only one or two bytes. Virtually all instructions execute in either one or two instruction cycles—one or two microseconds with a 12-MHz crystal—with the sole exceptions (multiply and divide) completing in four cycles.

Many instructions such as arithmetic and logical functions or program control, provide both a short and a long form for the same operation, allowing the programmer to optimize the code produced for a specific application. The 8051 usually fetches two instruction bytes per instruction cycle, so using a shorter form can lead to faster execution as well.

For example, any byte of RAM may be loaded with a constant with a three-byte, two-cycle instruction, but the commonly used “working registers” in RAM may be initialized in one cycle with a two-byte form. Any bit anywhere on the chip may be set, cleared, or complemented by a single three-byte logical instruction using two cycles. But critical control bits, I/O pins, and software flags may be controlled by two-byte, single cycle instructions. While three-byte jumps and calls can “go anywhere” in program memory, nearby sections of code may be reached by shorter relative or absolute versions.

AFN-01502A-09

| (MSB) |    |    |     | (LSB) |    |   |   |
|-------|----|----|-----|-------|----|---|---|
| CY    | AC | F0 | RS1 | RS0   | OV | — | P |

| Symbol | Position | Name and Significance  |
|--------|----------|--|
| CY     | PSW.7    | Carry flag.<br>Set/cleared by hardware or software during certain arithmetic and logical instructions.                                 |
| AC     | PSW.6    | Auxiliary Carry flag.<br>Set/cleared by hardware during addition or subtraction instructions to indicate carry or borrow out of bit 3. |
| F0     | PSW.5    | Flag 0<br>Set/cleared/tested by software as a user-defined status flag.  |
| RS1    | PSW.4    | Register bank Select control bits 1 & 0.<br>Set/cleared by software to determine working register bank (see Note).                     |
| RS     | PSW.3    |  |

| Symbol | Position | Name and Significance  |
|--------|----------|--|
| OV     | PSW.2    | Overflow flag.<br>Set/cleared by hardware during arithmetic instructions to indicate overflow conditions.  |
| ---    | PSW.1    | (reserved)   |
| P      | PSW.0    | Parity flag.<br>Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the accumulator, i.e., even parity. |

**Note—** the contents of (RS1, RS0) enable the working register banks as follows:

|                 |           |
|-----------------|-----------|
| (0,0) -- Bank 0 | (00H-07H) |
| (0,1) -- Bank 1 | (08H-0FH) |
| (1,0) -- Bank 2 | (10H-17H) |
| (1,1) -- Bank 3 | (18H-1FH) |

Figure 4. PSW—Program Status Word Organization

A significant side benefit of an instruction set more powerful than those of previous single-chip microcomputers is that it is easier to generate applications-oriented software. Generalized addressing modes for byte and bit instructions reduce the number of source code lines written and debugged for a given application. This leads in turn to proportionately lower software costs, greater reliability, and faster design cycles.

#### Accumulator and PSW

The 8051, like its 8048 predecessor, is primarily an accumulator-based architecture: an eight-bit register called the accumulator ("A") holds a source operand and receives the result of the arithmetic instructions (addition, subtraction, multiplication, and division). The accumulator can be the source or destination for logical operations and a number of special data movement instructions, including table look-ups and external RAM expansion. Several functions apply exclusively to the accumulator: rotates, parity computation, testing for zero, and so on.

Many instructions implicitly or explicitly affect (or are affected by) several status flags, which are grouped together to form the Program Status Word shown in Figure 4.

(The period within entries under the Position column is called the "dot operator," and indicates a particular bit position within an eight-bit byte. "PSW.5" specifies bit 5 of the PSW. Both the documentation and ASM51 use this notation.)

The most "active" status bit is called the carry flag (abbreviated "C"). This bit makes possible multiple precision arithmetic operations including addition, subtraction,

and rotates. The carry also serves as a "Boolean accumulator" for one-bit logical operations and bit manipulation instructions. The overflow flag (OV) detects when arithmetic overflow occurs on signed integer operands, making two's complement arithmetic possible. The parity flag (P) is updated after every instruction cycle with the even-parity of the accumulator contents.

The CPU does not control the two register-bank select bits, RS1 and RS0. Rather, they are manipulated by software to enable one of the four register banks. The usage of the PSW flags is demonstrated in the Instruction Set chapter of this Note.

Even though the architecture is accumulator-based, provisions have been made to bypass the accumulator in common instruction situations. Data may be moved from any location on-chip to any register, address, or indirect address (and vice versa), any register may be loaded with a constant, etc., all without affecting the accumulator. Logical operations may be performed against registers or variables to alter fields of bits—without using or affecting the accumulator. Variables may be incremented, decremented, or tested without using the accumulator. Flags and control bits may be manipulated and tested without affecting anything else.

#### Other CPU Registers

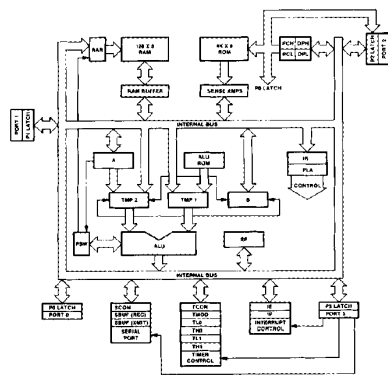
A special eight-bit register ("B") serves in the execution of the multiply and divide instructions. This register is used in conjunction with the accumulator as the second input operand and to return eight-bits of the result.

The MCS-51 family processors include a hardware stack within internal RAM, useful for subroutine linkage,

AFN-01502A-10

passing parameters between routines, temporary variable storage, or saving status during interrupt service routines. The Stack Pointer (SP) is an eight-bit pointer register which indicates the address of the last byte pushed onto the stack. The stack pointer is automatically incremented or decremented on all push or pop instructions and all subroutine calls and returns. In theory, the stack in the 8051 may be up to a full 128 bytes deep. (In practice, even simple programs would use a handful of RAM locations for pointers, variables, and so forth—reducing the stack depth by that number.) The stack pointer defaults to 7 on reset, so that the stack will start growing up from location 8, just like in the 8048. By altering the pointer contents the stack may be relocated anywhere within internal RAM.

Finally, a 16-bit register called the data pointer (DPTR) serves as a base register in indirect jumps, table look-up instructions, and external data transfers. The high- and low-order halves of the data pointer may be manipulated as separate registers (DPH and DPL, respectively) or together using special instructions to load or increment all sixteen bits. Unlike the 8048, look-up tables can therefore start anywhere in program memory and be of arbitrary length.



## Memory Spaces

Program memory is separate and distinct from data memory. Each memory type has a different addressing mechanism, different control signals, and a different function.

The program memory array (ROM or EPROM), like an elephant, is extremely large and never forgets information, even when power is removed. Program memory is used for information needed each time power is applied: initialization values, calibration constants, keyboard layout tables, etc., as well as the program itself. The program memory has a sixteen-bit address bus; its elements

are addressed using the Program Counter or instructions which generate a sixteen-bit address.

To stretch our analogy just a bit, data memory is like a mouse: it is smaller and therefore quicker than program memory, and it goes into a random state when electrical power is applied. On-chip data RAM is used for variables which are determined or may change while the program is running.

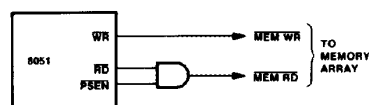
A computer spends most of its time manipulating variables, not constants, and a relatively small number of variables at that. Since eight-bits is more than sufficient to uniquely address 128 RAM locations, the on-chip RAM address register is only one byte wide. In contrast to the program memory, data memory accesses need a single eight-bit value—a constant or another variable—to specify a unique location. Since this is the basic width of the ALU and the different memory types, those resources can be used by the addressing mechanisms, contributing greatly to the computer's operating efficiency.

The partitioning of program and data memory is extended to off-chip memory expansion. Each may be added independently, and each uses the same address and data busses, but with different control signals. External program memory is gated onto the external data bus by the PSEN (Program Store Enable) control output, pin 29. External data memory is read onto the bus by the RD output, pin 17, and written with data supplied from the microcomputer by the WR output, pin 16. (There is no control pin to write external program ROM, which is by definition Read Only.) While both types may be expanded to up to 64K bytes, the external data memory may optionally be expanded in 256 byte "pages" to preserve the use of P2 as an I/O port. This is useful with a relatively small expansion RAM (such as the Intel® 8155) or for addressing external peripherals.

Single-chip controller programs are finalized during the project design cycle, and are not modified after production. Intel's single-chip microcomputers are not "von Neumann" architectures common among main-frame and mini-computer systems: the MCS-51™ processor data memory—on-chip and external—may *not* be used for program code. Just as there is no write-control signal for program memory, there is no way for the CPU to execute instructions out of RAM. In return, this concession allows an architecture optimized for efficient controller applications: a large, fixed program located in ROM, a hundred or so variables in RAM, and different methods for efficiently addressing each.

(Von Neumann machines are helpful for software development and debug. An 8051 system could be modified to have a single off-chip memory space by gating together the two memory-read controls (PSEN and RD) with a two-input AND gate (Figure 5). The CPU could then write data into the common memory array using WR and

AFN-01502A-11



**Figure 5. Combining External Program and Data Memory Arrays**

external data transfer instructions, and read instructions or data with the AND gate output and data transfer or program memory look-up instructions.)

In addition to the memory arrays, there is (yet) another (albeit sparsely populated) physical address space. Connected to the internal data bus are a score of special-purpose eight-bit registers scattered throughout the chip. Some of these—B, SP, PSW, DPH, and DPL—have been discussed above. Others—I/O ports and peripheral function registers—will be introduced in the following sections. Collectively, these registers are designated as the “special-function register” address space. Even the accumulator is assigned a spot in the special-function register address space for additional flexibility and uniformity.

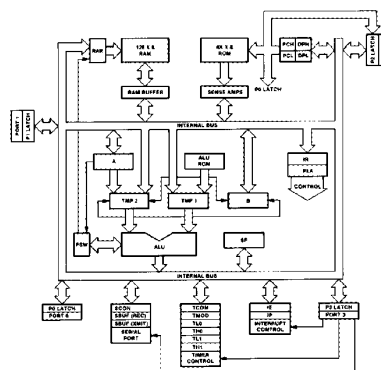
Thus, the MCS-51™ architecture supports several distinct “physical” address spaces, functionally separated at the hardware level by different addressing mechanisms, read and write control signals, or both:

- On-chip program memory;
- On-chip data memory;
- Off-chip program memory;
- Off-chip data memory;
- On-chip special-function registers.

What the *programmer sees*, though, are “logical” address spaces. For example, as far as the programmer is concerned, there is only one type of program memory, 64K bytes in length. The fact that it is formed by combining on- and off-chip arrays (split 4K/60K on the 8051 and 8751) is “invisible” to the programmer; the CPU automatically fetches each byte from the appropriate array, based on its address.

(Presumably, future microcomputers based on the MCS-51™ architecture may have a different physical split, with more or less of the 64K total implemented on-chip. Using the MCS-48™ family as a precedent, the 8048's 4K potential program address space was split 1K/3K between on- and off-chip arrays; the 8049's was split 2K/2K.)

Why go into such tedious details about address spaces? The logical addressing modes are described in the Instruction Set chapter in terms of physical address spaces. Understanding their differences now will pay off in understanding and using the chips later.



### Input/Output Ports

The MCS-51™ I/O port structure is extremely versatile. The 8051 and 8751 each have 32 I/O pins configured as four eight-bit parallel ports (P0, P1, P2, and P3). Each pin will input or output data (or both) under software control, and each may be referenced by a wide repertoire of byte and bit operations.

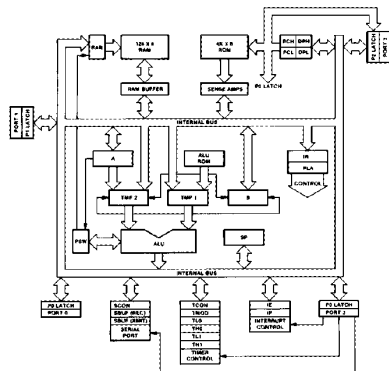
In various operating or expansion modes, some of these I/O pins are also used for special input or output functions. Instructions which access external memory use Port 0 as a multiplexed address/data bus: at the beginning of an external memory cycle eight bits of the address are output on P0; later data is transferred on the same eight pins. External data transfer instructions which supply a sixteen-bit address, and any instruction accessing external program memory, output the high-order eight bits on P2 during the access cycle. (The 8031 *always* uses the pins of P0 and P2 for external addressing, but P1 and P3 are available for standard I/O.)

The eight pins of Port 3 (P3) each have a special function. Two external interrupts, two counter inputs, two serial data lines, and two timing control strobes use pins of P3 as described in Figure 6. Port 3 pins corresponding to functions not used are available for conventional I/O.

Even within a single port, I/O functions may be combined in many ways: input and output may be performed using different pins at the same time, or the same pins at different times; in parallel in some cases, and in serial in others; as test pins, or (in the case of Port 3) as additional special functions.

| (MSB)         |                 |   |    | (LSB) |      |     |     |
|---------------|-----------------|---|----|-------|------|-----|-----|
| RD            | WR              | T1  | T0 | INT1  | INT0 | TXD | RXD |
| <b>Symbol</b> | <b>Position</b> | <b>Name and Significance</b>  |    |       |      |     |     |
| RD            | P3.7            | Read data control output. Active low pulse generated by hardware when external data memory is read.     |    |       |      |     |     |
| WR            | P3.6            | Write data control output. Active low pulse generated by hardware when external data memory is written. |    |       |      |     |     |
| T1            | P3.5            | Timer/counter 1 external input or test pin.   |    |       |      |     |     |
| T0            | P3.4            | Timer/counter 0 external input or test pin.   |    |       |      |     |     |
| INT1          | P3.3            | Interrupt 1 input pin. Low-level or falling-edge triggered.   |    |       |      |     |     |
| INT0          | P3.2            | Interrupt 0 input pin. Low-level or falling-edge triggered.   |    |       |      |     |     |
| TXD           | P3.1            | Transmit Data pin for serial port in UART mode. Clock output in shift register mode.                    |    |       |      |     |     |
| RXD           | P3.0            | Receive Data pin for serial port in UART mode. Data I/O pin in shift register mode.                     |    |       |      |     |     |

Figure 6. P3—Alternate Special Functions of Port 3



### Special Peripheral Functions

There are a few special needs common among control-oriented computer systems:

- keeping track of elapsed real-time;
- maintaining a count of signal transitions;
- measuring the precise width of input pulses;
- communicating with other systems or people;
- closely monitoring asynchronous external events.

Until now, microprocessor systems needed peripheral chips such as timer/counters, USARTs, or interrupt controllers to meet these needs. The 8051 integrates all of these capabilities on-chip!

#### Timer/Counters

There are two sixteen-bit multiple-mode Timer/Counters on the 8051, each consisting of a "High" byte (corresponding to the 8048 "T" register) and a low byte (similar to the 8048 prescaler, with the additional flexibility of being

software-accessible). These registers are called, naturally enough, TH0, TL0, TH1, and TL1. Each pair may be independently software programmed to any of a dozen modes with a mode register designated TMOD (Figure 7), and controlled with register TCON (Figure 8).

The timer modes can be used to measure time intervals, determine pulse widths, or initiate events, with one-micro-second resolution, up to a maximum interval of 65,536 instruction cycles (over 65 milliseconds). Longer delays may easily be accumulated through software. Configured as a counter, the same hardware will accumulate external events at frequencies from D.C. to 500 KHz, with up to sixteen bits of precision.

#### Serial Port Interface

Each microcomputer contains a high-speed, full-duplex, serial port which is software programmable to function in four basic modes: shift-register I/O expander, 8-bit UART, 9-bit UART, or interprocessor communications link. The UART modes will interface with standard I/O devices (e.g. CRTs, teletypewriters, or modems) at data rates from 122 baud to 31 kilobaud. Replacing the standard 12 MHz crystal with a 10.7 MHz crystal allows 110 baud. Even or odd parity (if desired) can be included with simple bit-handling software routines. Inter-processor communications in distributed systems takes place at 187 kilobaud with hardware for automatic address/data message recognition. Simple TTL or CMOS shift registers provide low-cost I/O expansion at a super-fast 1 Megabaud. The serial port operating modes are controlled by the contents of register SCON (Figure 9).

#### Interrupt Capability and Control

(Interrupt capability is generally considered a CPU function. It is being introduced here since, from an applications point of view, interrupts relate more closely to peripheral and system interfacing.)

AFN-01502A-13

| (MSB)   |     |    |    | (LSB)   |     |    |    | M1 | M0 | Operating Mode   |
|---------|-----|----|----|---------|-----|----|----|----|----|--|
| GATE    | C/T | M1 | M0 | GATE    | C/T | M1 | M0 |    |    |  |
| TIMER 1 |     |    |    | TIMER 0 |     |    |    | 0  | 0  | MCS-48 Timer. "TLx" serves as five-bit prescaler.  |
| GATE    |     |    |    |         |     |    |    | 0  | 1  | 16-bit timer-counter. "THx" and "TLx" are cascaded; there is no prescaler.   |
|         |     |    |    |         |     |    |    | 1  | 0  | 8-bit auto-reload timer counter. "THx" holds a value which is to be reloaded into "TLx" each time it overflows.  |
|         |     |    |    |         |     |    |    | 1  | 1  | (Timer 0) TL0 is an eight-bit timer counter controlled by the standard Timer 0 control bits.<br>TH0 is an eight-bit timer only controlled by Timer 1 control bits. |
| C/T     |     |    |    |         |     |    |    | 1  | 1  | (Timer 1) Timer counter 1 stopped.   |

Figure 7. TMOD—Timer/Counter Mode Register

| (MSB) |     |     |     | (LSB) |     |     |     |
|-------|-----|-----|-----|-------|-----|-----|-----|
| TF1   | TR1 | TF0 | TR0 | IE1   | IT1 | IE0 | IT0 |

| Symbol | Position | Name and Significance   |
|--------|----------|---|
| TF1    | TCON.7   | Timer 1 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed. |
| TR1    | TCON.6   | Timer 1 Run control bit. Set/cleared by software to turn timer/counter on/off.                      |
| TF0    | TCON.5   | Timer 0 overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed. |
| TR0    | TCON.4   | Timer 0 Run control bit. Set/cleared by software to turn timer/counter on/off.                      |

| Symbol | Position | Name and Significance  |
|--------|----------|--|
| IE1    | TCON.3   | Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.        |
| IT1    | TCON.2   | Interrupt 1 Type control bit. Set/cleared by software to specify falling edge low level triggered external interrupts. |
| IE0    | TCON.1   | Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.        |
| IT0    | TCON.0   | Interrupt 0 Type control bit. Set/cleared by software to specify falling edge low level triggered external interrupts. |

Figure 8. TCON—Timer/Counter Control/Status Register

| (MSB)         |                 |  |     | (LSB) |     |    |    |
|---------------|-----------------|--|-----|-------|-----|----|----|
| SM0           | SM1             | SM2  | REN | TB8   | RB8 | TI | RI |
| <b>Symbol</b> | <b>Position</b> | <b>Name and Significance</b>   |     |       |     |    |    |
| SM0           | SCON.7          | Serial port Mode control bit 0. Set/cleared by software (see note).  |     |       |     |    |    |
| SM1           | SCON.6          | Serial port Mode control bit 1. Set/cleared by software (see note).  |     |       |     |    |    |
| SM2           | SCON.5          | Serial port Mode control bit 2. Set by software to disable reception of frames for which bit 8 is zero.  |     |       |     |    |    |
| REN           | SCON.4          | Receiver Enable control bit. Set/cleared by software to enable/disable serial data reception.  |     |       |     |    |    |
| TB8           | SCON.3          | Transmit Bit 8. Set/cleared by hardware to determine state of ninth data bit transmitted in 9-bit UART mode.   |     |       |     |    |    |
| RB8           | SCON.2          | Receive Bit 8. Set/cleared by hardware to indicate state of ninth data bit received.   |     |       |     |    |    |
| TI            | SCON.1          | Transmit Interrupt flag. Set by hardware when byte transmitted. Cleared by software after servicing.   |     |       |     |    |    |
| RI            | SCON.0          | Received Interrupt flag. Set by hardware when byte received. Cleared by software after servicing.  |     |       |     |    |    |
| <b>Note—</b>  |                 | the state of (SM0,SM1) selects:<br>(0,0)—Shift register I/O expansion.<br>(0,1)—8 bit UART, variable data rate.<br>(1,0)—9 bit UART, fixed data rate.<br>(1,1)—9 bit UART, variable data rate. |     |       |     |    |    |

Figure 9. SCON—Serial Port Control/Status Register

These peripheral functions allow special hardware to monitor real-time signal interfacing without bothering the CPU. For example, imagine serial data is arriving from one CRT while being transmitted to another, and one timer/counter is tallying high-speed input transitions while the other measures input pulse widths. During all of this the CPU is thinking about something else.

But how does the CPU know when a reception, transmission, count, or pulse is finished? The 8051 programmer can choose from three approaches.

TCON and SCON contain status bits set by the hardware when a timer overflows or a serial port operation is completed. The first technique reads the control register into the accumulator, tests the appropriate bit, and does a conditional branch based on the result. This "polling" scheme (typically a three-instruction sequence though additional instructions to save and restore the accumulator may sometimes be needed) will surely be familiar to programmers used to multi-chip microcomputer systems and peripheral controller chips. This process is rather cumbersome, especially when monitoring multiple peripherals.

As a second approach, the 8051 can perform a conditional branch based on the state of any control or status bit or input pin in a single instruction; a four instruction sequence could poll the four simultaneous happenings mentioned above in just eight microseconds.

Unfortunately, the CPU must still drop what it's doing to test these bits. A manager cannot do his own work well if he is continuously monitoring his subordinates; they should interrupt him (or her) only when they need attention or guidance. So it is with machines: ideally, the CPU would not have to worry about the peripherals until they require servicing. At that time, it would postpone the

background task long enough to handle the appropriate device, then return to the point where it left off.

This is the basis of the third and generally optimal solution, hardware interrupts. The 8051 has five interrupt sources: one from the serial port when a transmission or reception is complete, two from the timers when overflows occur, and two from input pins INT0 and INT1. Each source may be independently enabled or disabled to allow polling on some sources or at some times, and each may be classified as high or low priority. A high priority source can interrupt a low priority service routine; the manager's boss can interrupt conferences with subordinates. These options are selected by the interrupt enable and priority control registers, IE and IP (Figures 10 and 11).

Each source has a particular program memory address associated with it (Table 3), starting at 0003H (as in the 8048) and continuing at eight-byte intervals. When an event enabled for interrupts occurs the CPU automatically executes an internal subroutine call to the corresponding address. A user subroutine starting at this location (or jumped to from this location) then performs the instructions to service that particular source. After completing the interrupt service routine, execution returns to the background program.

Table 3. 8051 Interrupt Sources and Service Vectors

| Interrupt Source | Service Routine Starting Address |
|------------------|----------------------------------|
| (Reset)          | 0000H                            |
| External 0       | 0003H                            |
| Timer/Counter 0  | 000BH                            |
| External 1       | 0013H                            |
| Timer/Counter 1  | 001BH                            |
| Serial Port      | 0023H                            |

AFN-01502A-15



| (MSB) |   |   |    | (LSB) |     |     |     |
|-------|---|---|----|-------|-----|-----|-----|
| EA    | — | — | ES | ET1   | EX1 | ET0 | EX0 |

| Symbol | Position | Name and Significance   | Symbol | Position | Name and Significance  |
|--------|----------|---|--------|----------|--|
| EA     | IE.7     | Enable All control bit. Cleared by software to disable all interrupts, independent of the state of IE.4-IE.0. | EX1    | IE.2     | Enable External interrupt 1 control bit. Set/cleared by software to enable/disable interrupts from INT1. |
| —      | IE.6     | (reserved)  | ET0    | IE.1     | Enable Timer 0 control bit. Set/cleared by software to enable/disable interrupts from timer counter 0.   |
| —      | IE.5     | (reserved)  | EX0    | IE.0     | Enable External interrupt 0 control bit. Set/cleared by software to enable/disable interrupts from INT0. |
| ES     | IE.4     | Enable Serial port control bit. Set/cleared by software to enable/disable interrupts from TI or RI flags.     |        |          |  |
| ET1    | IE.3     | Enable Timer 1 control bit. Set/cleared by software to enable/disable interrupts from timer/counter 1.        |        |          |  |

Figure 10. IE—Interrupt Enable Register

| (MSB) |   |   |    | (LSB) |     |     |     |
|-------|---|---|----|-------|-----|-----|-----|
| —     | — | — | PS | PT1   | PX1 | PT0 | PX0 |

| Symbol | Position | Name and Significance  | Symbol | Position | Name and Significance  |
|--------|----------|--|--------|----------|--|
| —      | IP.7     | (reserved)   | PX1    | IP.2     | External interrupt 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT1. |
| —      | IP.6     | (reserved)   |        |          |  |
| —      | IP.5     | (reserved)   | PT0    | IP.1     | Timer 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer counter 0.   |
| PS     | IP.4     | Serial port Priority control bit. Set/cleared by software to specify high/low priority interrupts for Serial port. | PX0    | IP.0     | External interrupt 0 Priority control bit. Set/cleared by software to specify high/low priority interrupts for INT0. |
| PT1    | IP.3     | Timer 1 Priority control bit. Set/cleared by software to specify high/low priority interrupts for timer/counter 1. |        |          |  |

Figure 11. IP—Interrupt Priority Control Register

Table 4. MCS-51™ Instruction Set Description

| ARITHMETIC OPERATIONS |  |      |     | DATA TRANSFER (cont.)                              |  |      |     |
|-----------------------|--|------|-----|--|--|------|-----|
| Mnemonic              | Description                              | Byte | Cyc | Mnemonic   | Description  | Byte | Cyc |
| ADD A,Rn              | Add register to Accumulator              | 1    | 1   | MOVC A,@A+DPTR                                     | Move Code byte relative to DPTR to A   | 1    | 2   |
| ADD A,direct          | Add direct byte to Accumulator           | 2    | 1   | MOVC A,@A+PC                                       | Move Code byte relative to PC to A   | 1    | 2   |
| ADD A,@Ri             | Add indirect RAM to Accumulator          | 1    | 1   | MOVX A,@Ri   | Move External RAM (8-bit addr) to A  | 1    | 2   |
| ADD A,#data           | Add immediate data to Accumulator        | 2    | 1   | MOVX A,@DPTR                                       | Move External RAM (16-bit addr) to A   | 1    | 2   |
| ADDC A,Rn             | Add register to Accumulator with Carry   | 1    | 1   | MOVX @Ri,A   | Move A to External RAM (8-bit addr)  | 1    | 2   |
| ADDC A,direct         | Add direct byte to A with Carry flag     | 2    | 1   | MOVX @DPTR,A                                       | Move A to External RAM (16-bit addr)   | 1    | 2   |
| ADDC A,@Ri            | Add indirect RAM to A with Carry flag    | 1    | 1   | PUSH direct  | Push direct byte onto stack  | 1    | 2   |
| ADDC A,#data          | Add immediate data to A with Carry flag  | 2    | 1   | POP direct   | Pop direct byte from stack   | 1    | 2   |
| SUBB A,Rn             | Subtract register from A with Borrow     | 1    | 1   | XCH A,Rn   | Exchange register with Accumulator   | 1    | 1   |
| SUBB A,direct         | Subtract direct byte from A with Borrow  | 2    | 1   | XCH A,direct                                       | Exchange direct byte with Accumulator  | 2    | 1   |
| SUBB A,@Ri            | Subtract indirect RAM from A w. Borrow   | 1    | 1   | XCH A,@Ri  | Exchange indirect RAM with A   | 1    | 1   |
| SUBB A,#data          | Subtract immed. data from A w. Borrow    | 2    | 1   | XCHD A,@Ri   | Exchange low-order Digi ind. RAM w A   | 1    | 1   |
| INC A                 | Increment Accumulator                    | 1    | 1   | BOOLEAN VARIABLE MANIPULATION                      |  |      |     |
| INC Rn                | Increment register                       | 1    | 1   | Mnemonic   | Description  | Byte | Cyc |
| INC direct            | Increment direct byte                    | 2    | 1   | CLR C  | Clear Carry flag   | 1    | 1   |
| INC @Ri               | Increment indirect RAM                   | 1    | 1   | CLR bit  | Clear direct bit   | 2    | 1   |
| DEC A                 | Decrement Accumulator                    | 1    | 1   | SETB C   | Set Carry flag   | 1    | 1   |
| DEC Rn                | Decrement register                       | 1    | 1   | SETB bit   | Set direct bit   | 2    | 1   |
| DEC direct            | Decrement direct byte                    | 2    | 1   | CPL C  | Complement Carry flag  | 1    | 1   |
| DEC @Ri               | Decrement indirect RAM                   | 1    | 1   | CPL bit  | Complement direct bit  | 2    | 1   |
| INC DPTR              | Increment Data Pointer                   | 1    | 2   | ANL C,bit  | AND direct bit to Carry flag   | 2    | 2   |
| MUL AB                | Multiply A & B                           | 1    | 4   | ANL C,bit  | AND complement of direct bit to Carry  | 2    | 2   |
| DIV AB                | Divide A by B                            | 1    | 4   | ORI C,bit  | OR direct bit to Carry flag  | 2    | 2   |
| DA A                  | Decimal Adjust Accumulator               | 1    | 1   | ORI C,bit  | OR complement of direct bit to Carry   | 2    | 2   |
| LOGICAL OPERATIONS    |  |      |     | MOV C,bit  | Move direct bit to Carry flag  | 2    | 1   |
| Mnemonic              | Description                              | Byte | Cyc | MOV bit,C  | Move Carry flag to direct bit  | 2    | 2   |
| ANL A,Rn              | AND register to Accumulator              | 1    | 1   | PROGRAM AND MACHINE CONTROL                        |  |      |     |
| ANL A,direct          | AND direct byte to Accumulator           | 2    | 1   | Mnemonic   | Description  | Byte | Cyc |
| ANL A,@Ri             | AND indirect RAM to Accumulator          | 1    | 1   | ACALL addr11                                       | Absolute Subroutine Call   | 3    | 2   |
| ANL A,#data           | AND immediate data to Accumulator        | 2    | 1   | LCALL addr16                                       | Long Subroutine Call   | 3    | 2   |
| ANL direct,A          | AND Accumulator to direct byte           | 2    | 1   | RET  | Return from subroutine   | 1    | 2   |
| ANL direct,#data      | AND immediate data to direct byte        | 3    | 2   | RETI   | Return from interrupt  | 1    | 2   |
| ORL A,Rn              | OR register to Accumulator               | 1    | 1   | AJMP addr11  | Absolute Jump  | 2    | 2   |
| ORL A,direct          | OR direct byte to Accumulator            | 2    | 1   | LJMP addr16  | Long Jump  | 3    | 2   |
| ORL A,@Ri             | OR indirect RAM to Accumulator           | 1    | 1   | SJMP rel   | Short Jump (relative addr)   | 2    | 2   |
| ORL A,#data           | OR immediate data to Accumulator         | 2    | 1   | JMP @A+DPTR  | Jump indirect relative to the DPTR   | 1    | 2   |
| ORL direct,A          | OR Accumulator to direct byte            | 2    | 1   | JZ rel   | Jump if Accumulator is Zero  | 2    | 2   |
| ORL direct,#data      | OR immediate data to direct byte         | 3    | 2   | JNZ rel  | Jump if Accumulator is Not Zero  | 2    | 2   |
| XRL A,Rn              | Exclusive-OR register to Accumulator     | 1    | 1   | JC rel   | Jump if Carry flag is set  | 2    | 2   |
| XRL A,direct          | Exclusive-OR direct byte to Accumulator  | 2    | 1   | JNC rel  | Jump if No Carry flag  | 2    | 2   |
| XRL A,@Ri             | Exclusive-OR indirect RAM to A           | 1    | 1   | JB bit,rel   | Jump if direct Bit set   | 3    | 2   |
| XRL A,#data           | Exclusive-OR immediate data to A         | 2    | 1   | JNB bit,rel  | Jump if direct Bit Not set   | 3    | 2   |
| XRL direct,A          | Exclusive-OR Accumulator to direct byte  | 2    | 1   | JBC bit,rel  | Jump if direct Bit is set & Clear bit  | 3    | 2   |
| XRL direct,#data      | Exclusive-OR immediate data to direct    | 3    | 2   | CJNE A,direct,rel                                  | Compare direct to A & Jump if Not Equal  | 3    | 2   |
| CLR A                 | Clear Accumulator                        | 1    | 1   | CJNE Rn,#data,rel                                  | Comp. immed. to reg. & Jump if Not Equal   | 3    | 2   |
| CPL A                 | Complement Accumulator                   | 1    | 1   | CJNE @Ri,#data,rel                                 | Comp. immed. to ind. & Jump if Not Equal   | 3    | 2   |
| RL A                  | Rotate Accumulator Left                  | 1    | 1   | DJNZ Rn,rel  | Decrement register & Jump if Not Zero  | 2    | 2   |
| RLC A                 | Rotate A Left through the Carry flag     | 1    | 1   | DJNZ direct,rel                                    | Decrement direct & Jump if Not Zero  | 3    | 2   |
| RR A                  | Rotate Accumulator Right                 | 1    | 1   | NOP  | No operation   | 1    | 1   |
| RRC A                 | Rotate A Right through Carry flag        | 1    | 1   | Notes on data addressing modes:                    |  |      |     |
| SWAP A                | Swap nibbles within the Accumulator      | 1    | 1   | Rn   | Working register R0-R7   |      |     |
| DATA TRANSFER         |  |      |     | direct   | 128 internal RAM locations, any I/O port, control or status register   |      |     |
| Mnemonic              | Description                              | Byte | Cyc | @Ri  | Indirect internal RAM location addressed by register R0 or R1  |      |     |
| MOV A,Rn              | Move register to Accumulator             | 1    | 1   | #data  | 8-bit constant included in instruction   |      |     |
| MOV A,direct          | Move direct byte to Accumulator          | 2    | 1   | #data16  | 16-bit constant included as bytes 2 & 3 of instruction   |      |     |
| MOV A,@Ri             | Move indirect RAM to Accumulator         | 1    | 1   | bit  | 128 software flags, any I/O pin, control or status bit   |      |     |
| MOV A,#data           | Move immediate data to Accumulator       | 2    | 1   | Notes on program addressing modes:                 |  |      |     |
| MOV Rn,A              | Move Accumulator to register             | 1    | 1   | addr16   | Destination address for LCALL & LJMP may be anywhere within the 64-Kilobyte program memory address space.                                      |      |     |
| MOV Rn,direct         | Move direct byte to register             | 2    | 2   | addr11   | Destination address for ACALL & AJMP will be within the same 2-Kilobyte page of program memory as the first byte of the following instruction. |      |     |
| MOV Rn,#data          | Move immediate data to register          | 2    | 1   | rel  | SJMP and all conditional jumps include an 8-bit offset byte. Range is +127 -128 bytes relative to first byte of the following instruction.     |      |     |
| MOV direct,A          | Move Accumulator to direct byte          | 2    | 1   | All mnemonics copyrighted © Intel Corporation 1979 |  |      |     |
| MOV direct,Rn         | Move register to direct byte             | 2    | 2   |  |  |      |     |
| MOV direct,direct     | Move direct byte to direct               | 3    | 2   |  |  |      |     |
| MOV direct,@Ri        | Move indirect RAM to direct byte         | 2    | 2   |  |  |      |     |
| MOV direct,#data      | Move immediate data to direct byte       | 3    | 2   |  |  |      |     |
| MOV @Ri,A             | Move Accumulator to indirect RAM         | 1    | 1   |  |  |      |     |
| MOV @Ri,direct        | Move direct byte to indirect RAM         | 2    | 2   |  |  |      |     |
| MOV @Ri,#data         | Move immediate data to indirect RAM      | 2    | 1   |  |  |      |     |
| MOV DPTR,#data16      | Load Data Pointer with a 16-bit constant | 3    | 2   |  |  |      |     |

### 3. INSTRUCTION SET AND ADDRESSING MODES

The 8051 instruction set is extremely regular, in the sense that most instructions can operate with variables from several different physical or logical address spaces. Before getting deeply enmeshed in the instruction set proper, it is important to understand the details of the most common data addressing modes. Whereas Table 4 summarizes the instructions set broken down by functional

group, this chapter starts with the addressing mode classes and builds to include the related instructions.

#### Data Addressing Modes

MCS-51 assembly language instructions consist of an operation mnemonic and zero to three operands separated by commas. In two operand instructions the destination is specified first, then the source. Many byte-wide data

AFN-01502A-17

operations (such as ADD or MOV) inherently use the accumulator as a source operand and/or to receive the result. For the sake of clarity the letter "A" is specified in the source or destination field in all such instructions. For example, the instruction,

```
ADD A,<source>
```

will add the variable<source>to the accumulator, leaving the sum in the accumulator.

The operand designated "<source>" above may use any of four common logical addressing modes:

- Register—one of the working registers in the currently enabled bank.
- Direct—an internal RAM location, I/O port, or special-function register.
- Register-indirect—an internal RAM location, pointed to by a working register.
- Immediate data—an eight-bit constant incorporated into the instruction.

The first three modes provide access to the internal RAM and Hardware Register address spaces, and may therefore be used as source or destination operands; the last mode accesses program memory and may be a source operand only.

(It is hard to show a "typical application" of any instruction without involving instructions not yet described. The following descriptions use only the self-explanatory ADD and MOV instructions to demonstrate how the four addressing modes are specified and used. Subsequent examples will become increasingly complex.)

#### Register Addressing

The 8051 programmer has access to eight "working registers," numbered R0-R7. The least-significant three-bits of the instruction opcode indicate one register within this logical address space. Thus, a function code and operand address can be combined to form a short (one byte) instruction (Figure 12.a).

The 8051 assembly language indicates register addressing with the symbol Rn (where n is from 0 to 7) or with a symbolic name previously defined as a register by the EQUate or SET directives. (For more information on assembler directives see the Macro Assembler Reference Manual.)

#### Example 1—Adding Two Registers Together

```
REGADR ADD CONTENTS OF REGISTER 1
      TO CONTENTS OF REGISTER 0
REGADR MOV A,R0
      ADD A,R1
      MOV R0,A
```

There are four such banks of working registers, only one of which is active at a time. Physically, they occupy the first 32 bytes of on-chip data RAM (addresses 0-1FH). PSW bits 4 and 3 determine which bank is active. A

hardware reset enables register bank 0; to select a different bank the programmer modifies PSW bits 4 and 3 accordingly.

#### Example 2—Selecting Alternate Memory Banks

```
MOV PSW,#00010000B ;SELECT BANK 2
```

Register addressing in the 8051 is the same as in the 8048 family, with two enhancements: there are four banks rather than one or two, and 16 instructions (rather than 12) can access them.

#### Direct Byte Addressing

Direct addressing can access any on-chip variable or hardware register. An additional byte appended to the opcode specifies the location to be used (Figure 12.b).

Depending on the highest order bit of the direct addressing byte, one of two physical memory spaces is selected. When the direct address is between 0 and 127 (00H-7FH) one of the 128 low-order on-chip RAM locations is used. (Future microcomputers based on the MCS-51™ architecture may incorporate more than 128 bytes of on-chip RAM. Even if this is the case, only the low-order 128 bytes will be directly addressable. The remainder would be accessed indirectly or via the stack pointer.)

#### Example 3—Adding RAM Location Contents

```
DIRADR ADD CONTENTS OF RAM LOCATION 41H
      TO CONTENTS OF RAM LOCATION 40H
DIRADR MOV A,40H
      ADD A,41H
      MOV 40H,A
```

All I/O ports and special function, control, or status registers are assigned addresses between 128 and 255 (80H-0FFH). When the direct address byte is between these limits the corresponding hardware register is accessed. For example, Ports 0 and 1 are assigned direct addresses 80H and 90H, respectively. A complete list is presented in Table 5. Don't waste your time trying to memorize the addresses in Table 5. Since programs using absolute addresses for function registers would be difficult to write or understand, ASM51 allows and understands the abbreviations listed instead.

#### Example 4—Adding Input Port Data to Output Port Data

```
PRTADR ADD DATA INPUT ON PORT 1
      TO DATA PREVIOUSLY OUTPUT
      ON PORT 0
PRTADR MOV A,P0
      ADD A,P1
      MOV P0,A
```

Direct addressing allows all special-function registers in the 8051 to be read, written, or used as instruction operands. In general, this is the *only* method used for accessing I/O ports and special-function registers. If direct addressing is used with special-function register addresses other than those listed, the result of the instruction is undefined.

AFN-01502A-18

The 8048 does not have or need any generalized direct addressing mode, since there are only five special registers (BUS, P1, P2, PSW, & T) rather than twenty. Instead, 16 special 8048 opcodes control output bits or read or write each register to the accumulator. These functions are all subsumed by four of the 27 direct addressing instructions of the 8051.

Table 5. 8051 Hardware Register Direct Addresses

| Register | Address | Function                     |
|----------|---------|------------------------------|
| P0       | 80H*    | Port 0                       |
| SP       | 81H     | Stack Pointer                |
| DPL      | 82H     | Data Pointer (Low)           |
| DPH      | 83H     | Data Pointer (High)          |
| TCON     | 88H*    | Timer register               |
| TMOD     | 89H     | Timer Mode register          |
| TL0      | 8AH     | Timer 0 Low byte             |
| TL1      | 8BH     | Timer 1 Low byte             |
| TH0      | 8CH     | Timer 0 High byte            |
| TH1      | 8DH     | Timer 1 High byte            |
| P1       | 90H*    | Port 1                       |
| SCON     | 98H*    | Serial Port Control register |
| SBUF     | 99H     | Serial Port data Buffer      |
| P2       | 0A0H*   | Port 2                       |
| IE       | 0A8H*   | Interrupt Enable register    |
| P3       | 0B0H*   | Port 3                       |
| IP       | 0B8H*   | Interrupt Priority register  |
| PSW      | 0D0H*   | Program Status Word          |
| ACC      | 0E0H*   | Accumulator (direct address) |
| B        | 0F0H*   | B register                   |

\* = bit addressable register.

#### Register-Indirect Addressing

How can you handle variables whose locations in RAM are determined, computed, or modified while the program is running? This situation arises when manipulating sequential memory locations, indexed entries within tables in RAM, and multiple precision or string operations. Register or Direct addressing cannot be used, since their operand addresses are fixed at assembly time.

The 8051 solution is "register-indirect RAM addressing." R0 and R1 of each register bank may operate as index or pointer registers, their contents indicating an address into RAM. The internal RAM location so addressed is the actual operand used. The least significant bit of the instruction opcode determines which register is used as the "pointer" (Figure 12.c).

In the 8051 assembly language, register-indirect addressing is represented by a commercial "at" sign ("@" ) preceding R0, R1, or a symbol defined by the user to be equal to R0 or R1.

#### Example 5—Indirect Addressing

```

; INDR0 ADD CONTENTS OF MEMORY LOCATION
;   ADDRESSED BY REGISTER 1
; TO CONTENTS OF RAM LOCATION
;   ADDRESSED BY REGISTER 0
INDR0  MOV  A, R0
ADD    A, R1
MOV    R0, A

```

Indirect addressing on the 8051 is the same as in the 8048 family, except that all eight bits of the pointer register contents are significant; if the contents point to a non-existent memory location (i.e., an address greater than 7FH on the 8051) the result of the instruction is undefined. (Future microcomputers based on the MCS-51™ architecture could implement additional memory in the on-chip RAM logical address space at locations above 7FH.) The 8051 uses register-indirect addressing for five new instructions plus the 13 on the 8048.

#### Immediate Addressing

When a source operand is a constant rather than a variable (i.e., the instruction uses a value known at assembly time), then the constant can be incorporated into the instruction. An additional instruction byte specifies the value used (Figure 12.d).

The value used is fixed at the time of ROM manufacture or EPROM programming and may not be altered during program execution. In the assembly language immediate operands are preceded by a number sign ("#"). The operand may be either a numeric string, a symbolic variable, or an arithmetic expression using constants.

#### Example 6—Adding Constants Using Immediate Addressing

```

; IMADR ADD THE CONSTANT 12 (DECIMAL)
; TO THE CONSTANT 34 (DECIMAL)
; LEAVE SUM IN ACCUMULATOR
IMADR  MOV  A, #12
ADD    A, #34

```

The preceding example was included for consistency; it has little practical value. Instead, ASM51 could compute the sum of two constants at assembly time.

#### Example 7—Adding Constants Using ASM51 Capabilities

```

; ASMSUM LOAD ACC WITH THE SUM OF
; THE CONSTANT 12 (DECIMAL) AND
; THE CONSTANT 34 (DECIMAL)
ASMSUM MOV  A, # (12+34)

```

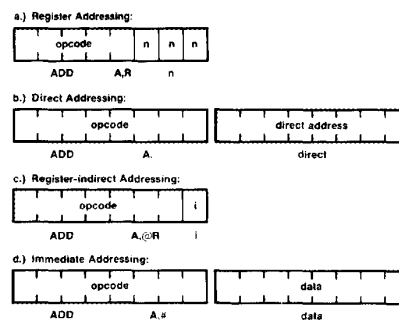


Figure 12. Data Addressing Machine Code Formats

AFN-01502A-19

### Addressing Mode Combinations

The above examples all demonstrated the use of the four data-addressing modes in two-operand instructions (MOV, ADD) which use the accumulator as one operand. The operations ADDC, SUBB, ANL, ORL, and XRL (all to be discussed later) could be substituted for ADD in each example. The first three modes may be also be used for the XCH operation or, in combination with the Immediate Addressing mode (and an additional byte), loaded with a constant. The one-operand instructions INC and DEC, DJNZ, and CJNE may all operate on the accumulator, or may specify the Register, Direct, and Register-indirect addressing modes. Exception: as in the 8048, DJNZ cannot use the accumulator or indirect addressing. (The PUSH and POP operations cannot inherently address the accumulator as a special register either. However, all three can *directly* address the accumulator as one of the twenty special-function registers by putting the symbol "ACC" in the operand field.)

### Advantages of Symbolic Addressing

Like most assembly or higher-level programming languages, ASM51 allows instructions or variables to be given appropriate, user-defined symbolic names. This is done for instruction lines by putting a label followed by a colon (":") before the instruction proper, as in the above examples. Such symbols must start with an alphabetic character (remember what distinguished BACH from 0BACH?), and may include any combination of letters, numbers, question marks ("?",) and underscores ("\_"). For **very** long names only the first 31 characters are relevant.

Assembly language programs may intermix upper- and lower-case letters arbitrarily, but ASM51 converts both to upper-case. For example, ASM51 will internally process an "I" for an "i" and, of course, "A\_TOOTH" for "a\_tooth."

The underscore character makes symbols easier to read and can eliminate potential ambiguity (as in the label for a subroutine to switch two entires on a stack, "S\_EXCHANGE"). The underscore is significant, and would distinguish between otherwise-identical character strings.

ASM51 allows *all* variables (registers, ports, internal or external RAM addresses, constants, etc.) to be assigned labels according to these rules with the EQUate or SET directives.

#### Example 8 Symbolic Addressing of Variables Defined as RAM Locations

```
VAR_0 SET 20H
VAR_1 SET 21H
SYMB_1 ADD CONTENTS OF VAR_1
      TO CONTENTS OF VAR_0
SYMB_1 MOV A,VAR_0
      ADD A,VAR_1
      MOV VAR_0,A
```

Notice from Table 4 that the MCS-51™ instruction set has relatively few instruction mnemonics (abbreviations) for the programmer to memorize. Different data types or addressing modes are determined by the operands specified, rather than variations on the mnemonic. For example, the mnemonic "MOV" is used by 18 different instructions to operate on three data types (bit, byte, and address). The fifteen versions which move byte variables between the logical address spaces are diagrammed in Figure 13. Each arrow shows the direction of transfer from source to destination.

Notice also that for most instructions allowing register addressing there is a corresponding direct addressing instruction and vice versa. This lets the programmer begin writing 8051 programs as if (s)he has access to 128 different registers. When the program has evolved to the point where the programmer has a fairly accurate idea how often each variable is used, he/she may allocate the working registers in each bank to the most "popular" variables. (The assembly cross-reference option will show exactly how often and where each symbol is referenced.) If symbolic addressing is used in writing the source program only the lines containing the symbol definition will need to be changed; the assembler will produce the appropriate instructions even though the rest of the program is left untouched. Editing only the first two lines of Example 8 will shrink the six-byte code segment produced in half.

How are instruction sets "counted"? There is no standard practice; different people assessing the same CPU using different conventions may arrive at different totals.

Each operation is then broken down according to the different addressing modes (or combinations of addressing modes) it can accommodate. The "CLR" mnemonic is used by two instructions with respect to bit variables ("CLR C" and "CLR bit") and once ("CLR A") with regards to bytes. This expansion yields the 111 separate instructions of Table 4.

The method used for the MCS-51\* instruction set first breaks it down into "operations": a basic function applied to a single data type. For example, the four versions of the ADD instruction are grouped to form one operation — addition of eight-bit variables. The six forms of the ANL instruction for *byte* variables make up a different operation; the two forms of ANL which operate on *bits* are considered still another. The MOV mnemonic is used by three different operation classes, depending on whether bit, byte, or 16-bit values are affected. Using this terminology the 8051 can perform 51 different operations.

AFN-01502A-20

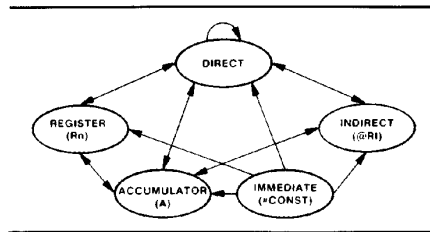


Figure 13. Road map for moving data bytes

Example 9—Redeclaring Example 8 Symbols as Registers

```

VAR_0 SET R0
VAR_1 SET R1
SYMB_2 ADD CONTENTS OF VAR_1
TO CONTENTS OF VAR_0
SYMB_2 MOV A, VAR_0
ADD A, VAR_1
MOV VAR_0, A
    
```

#### Arithmetic Instruction Usage — ADD, ADDC, SUBB and DA

The ADD instruction adds a byte variable with the accumulator, leaving the result in the accumulator. The carry flag is set if there is an overflow from bit 7 and cleared otherwise. The AC flag is set to the carry-out from bit 3 for use by the DA instruction described later. ADDC adds the previous contents of the carry flag with the two byte variables, but otherwise is the same as ADD.

The SUBB (subtract with borrow) instruction subtracts the byte variable indicated and the contents of the carry flag together from the accumulator, and puts the result back in the accumulator. The carry flag serves as a "Borrow Required" flag during subtraction operations; when a greater value is subtracted from a lesser value (as in subtracting 5 from 1) requiring a borrow into the highest order bit, the carry flag is set; otherwise it is cleared.

When performing signed binary arithmetic, certain combinations of input variables can produce results which seem to violate the Laws of Mathematics. For example, adding 7FH (127) to itself produces a sum of 0FEH, which is the two's complement representation of -2 (refer back to Table 2)! In "normal" arithmetic, two positive values can't have a negative sum. Similarly, it is normally impossible to subtract a positive value from a negative value and leave a positive result—but in two's complement there are instances where this too may happen. Fundamentally, such anomalies occur when the magnitude of the resulting value is too great to "fit" into the seven bits allowed for it; there is no one-byte two's complement representation for 254, the true sum of 127 and 127.

The MCS-51™ processors detect whether these situations occur and indicate such errors with the OV flag. (OV may be tested with the conditional jump instructions JB and JNB, described under the Boolean Processor chapter.)

At a hardware level, OV is set if there is a carry out of bit 6 but not out of bit 7, or a carry out of bit 7 but not out of bit 6. When adding signed integers this indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands; on SUBB this indicates a negative result after subtracting a negative number from a positive number, or a positive result when a positive number is subtracted from a negative number.

The ADDC and SUBB instructions incorporate the previous state of the carry (borrow) flag to allow multiple precision calculations by repeating the operation with successively higher-order operand bytes. In either case, the carry must be cleared before the first iteration.

If the input data for a multiple precision operation is an unsigned string of integers, upon completion the carry flag will be set if an overflow (for ADDC) or underflow (for SUBB) occurs. With two's complement signed data (i.e., if the most significant bit of the original input data indicates the sign of the string), the overflow flag will be set if overflow or underflow occurred.

Example 10—String Subtraction with Signed Overflow Detection

```

SUBSTR SUBTRACT STRING INDICATED BY R1
FROM STRING INDICATED BY R0 TO
PRECISION INDICATED BY R2
CHECK FOR SIGNED UNDERFLOW WHEN DONE

SUBSTR CLR C ;Borrow= 0
SUBS1 MOV A, @R0 ;SUBTRACT NEXT PLACE
SUBB A, @R1
MOV @R0, A
INC R0 ;BUMP POINTERS
INC R1
DJNZ R2, SUBS1 ;LOOP AS NEEDED
WHEN DONE, TEST IF OVERFLOW OCCURRED
ON LAST ITERATION OF LOOP
OV, OV, OK
JNB OV, OK ;OVERFLOW RECOVERY ROUTINE
RET ;RETURN
    
```

Decimal addition is possible by using the DA instruction in conjunction with ADD and/or ADDC. The eight-bit binary value in the accumulator resulting from an earlier addition of two variables (each a packed BCD digit-pair) is adjusted to form two BCD digits of four bits each. If the contents of accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag had been set, six is added to the accumulator producing the proper BCD digit in the low-order nibble. (This addition might itself set—but would not clear—the carry flag.) If the carry flag is set, or if the four high-order bits now exceed nine (1010xxxx-1111xxxx), these bits are incremented by six. The carry flag is left set if originally set or if either addition of six produces a carry out of the highest-order bit, indicating the sum of the original two BCD variables is greater than or equal to decimal 100.

AFN-01502A-21

### Example 11—Two Byte Decimal Add with Registers and Constants

```

BCDADD ADD THE CONSTANT 1,234 (DECIMAL) TO THE
        CONTENTS OF REGISTER PAIR (R3)(R2)
        (ALREADY A 4 BCD-DIGIT VARIABLE)
BCDADD MOV     A, R2
        ADD     A, #34H
        DA
        MOV     R2, A
        MOV     A, R3
        ADDC    A, #12H
        DA
        MOV     R3, A
        RET

```

### Multiplication and Division

The instruction "MUL AB" multiplies the unsigned eight-bit integer values held in the accumulator and B-registers. The low-order byte of the sixteen-bit product is left in the accumulator, the higher-order byte in B. If the high-order eight-bits of the product are all zero the overflow flag is cleared; otherwise it is set. The programmer can poll OV to determine when the B register is non-zero and must be processed.

"DIV AB" divides the unsigned eight-bit integer in the accumulator by the unsigned eight-bit integer in the B-register. The integer part of the quotient is returned in the accumulator; the remainder in the B-register. If the B-register originally contained 00H then the overflow flag will be set to indicate a division error, and the values returned will be undefined. Otherwise OV is cleared.

The divide instruction is also useful for purposes such as radix conversion or separating bit fields of the accumulator. A short subroutine can convert an eight-bit unsigned binary integer in the accumulator (between 0 & 255) to a three-digit (two byte) BCD representation. The hundred's digit is returned in one register (HUND) and the ten's and one's digits returned as packed BCD in another (TENONE).

### Example 12—Use of DIV Instruction for Radix Conversion

```

BINBCD CONVERT 8-BIT BINARY VARIABLE IN ACC
        TO 3-DIGIT PACKED BCD FORMAT
        HUNDREDS PLACE LEFT IN VARIABLE 'HUND',
        TENS' AND ONES' PLACES IN 'TENONE'
HUND EQU 21H
TENONE EQU 22H
BINBCD MOV     B, #100 ; DIVIDE BY 100 TO
        DIV     AB      ; DETERMINE NUMBER OF HUNDREDS
        MOV     HUND, A
        MOV     A, #10  ; DIVIDE REMAINDER BY 10 TO
        RCH     B        ; DETERMINE # OF TENS LEFT
        DIV     AB      ; TENS DIGIT IN ACC, REMAINDER IS ONES
        SHAP     A       ; DIGIT
        ADD     A, B     ; PACK BCD DIGITS IN ACC
        MOV     TENONE, A
        RET

```

The divide instruction can also separate eight bits of data in the accumulator into sub-fields. For example, packed BCD data may be separated into two nibbles by dividing the data by 16, leaving the high-nibble in the accumulator and the low-order nibble (remainder) in B. The two digits may then be operated on individually or in conjunction with each other. This example receives two packed BCD

digits in the accumulator and returns the product of the two individual digits in packed BCD format in the accumulator.

### Example 13—Implementing a BCD Multiply Using MPY and DIV

```

MULBCD UNPACK TWO BCD DIGITS RECEIVED IN ACC,
        FIND THEIR PRODUCT, AND RETURN PRODUCT
        IN PACKED BCD FORMAT IN ACC
MULBCD MOV     B, #10H ; DIVIDE INPUT BY 16
        DIV     AB      ; A & B HOLD SEPARATED DIGITS
        MUL     AB      ; (EACH RIGHT JUSTIFIED IN REGISTER)
        ; A HOLDS PRODUCT IN BINARY FORMAT (0 -
        ; 99(DECIMAL) = 0 - 63H)
        MOV     B, #10  ; DIVIDE PRODUCT BY 10
        DIV     AB      ; A HOLDS # OF TENS, B HOLDS REMAINDER
        SWAP    A        ;
        ORL     A, B     ; PACK DIGITS
        RET

```

### Logical Byte Operations — ANL, ORL, XRL

The instructions ANL, ORL, and XRL perform the logical functions AND, OR, and/or Exclusive-OR on the two byte variables indicated, leaving the results in the first. No flags are affected. (A word to the wise — do not vocalize the first two mnemonics in mixed company.)

These operations may use all the same addressing modes as the arithmetics (ADD, etc.) but unlike the arithmetics, they are not restricted to operating on the accumulator. Directly addressed bytes may be used as the destination with either the accumulator or a constant as the source. These instructions are useful for clearing (ANL), setting (ORL), or complementing (XRL) one or more bits in a RAM, output ports, or control registers. The pattern of bits to be affected is indicated by a suitable mask byte. Use immediate addressing when the pattern to be affected is known at assembly time (Figure 14); use the accumulator versions when the pattern is computed at run-time.

I/O ports are often used for parallel data in formats other than simple eight-bit bytes. For example, the low-order five bits of port 1 may output an alphabetic character code (hopefully) without disturbing bits 7-5. This can be a simple two-step process. First, clear the low-order five pins with an ANL instruction; then set those pins corresponding to ones in the accumulator. (This example assumes the three high-order bits of the accumulator are originally zero.)

### Example 14—Reconfiguring Port Size with Logical Byte Instructions

```

OUT_PX ANL P1, #11100000B ; CLEAR BITS P1 4 - P1 0
        ORL P1, A          ; SET P1 PINS CORRESPONDING TO SET ACC
        ; BITS
        RET

```

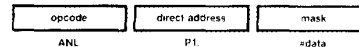


Figure 14. Instruction Pattern for Logical Operation Special Addressing Modes

AFN-01502A-22

In this example, low-order bits remaining high may “glitch” low for one machine cycle. If this is undesirable, use a slightly different approach. First, set all pins corresponding to accumulator one bits, then clear the pins corresponding to zeroes in low-order accumulator bits. Not all bits will change from original to final state at the same instant, but no bit makes an intermediate transition.

**Example 15 — Reconfiguring I/O Port Size without Glitching**

```
ALT_PX: ORL     P1, A
        ORL     A, #11100000B
        ANL     P1, A
        RET
```

**Program Control — Jumps, Calls, Returns**

Whereas the 8048 only has a single form of the simple jump instruction, the 8051 has three. Each causes the program to unconditionally jump to some other address. They differ in how the machine code represents the destination address.

LJMP (Long Jump) encodes a sixteen-bit address in the second and third instruction bytes (Figure 15.a); the destination may be anywhere in the 64 Kilobyte program memory address space.

The two-byte AJMP (Absolute Jump) instruction encodes its destination using the same format as the 8048: address bits 10 through 8 form a three bit field in the opcode and address bits 7 through 0 form the second byte (Figure 15.b). Address bits 15-12 are unchanged from the (incremented) contents of the P.C., so AJMP can only be used when the destination is known to be within the same 2K memory block. (Otherwise ASM51 will point out the error.)

A different two-byte jump instruction is legal with any nearby destination, regardless of memory block boundaries or “pages.” SJMP (Short Jump) encodes the destination with a program counter-relative address in the second byte (Figure 15.c). The CPU calculates the

destination at run-time by adding the signed eight-bit displacement value to the incremented P.C. Negative offset values will cause jumps up to 128 bytes backwards; positive values up to 127 bytes forwards. (SJMP with 00H in the machine code offset byte will proceed with the following instruction).

In keeping with the 8051 assembly language goal of minimizing the number of instruction mnemonics, there is a “generic” form of the three jump instructions. ASM51 recognizes the mnemonic JMP as a “pseudo-instruction,” translating it into the machine instructions LJMP, AJMP, or SJMP, depending on the destination address.

Like SJMP, all conditional jump instructions use relative addressing. JZ (Jump if Zero) and JNZ (Jump if Not Zero) monitor the state of the accumulator as implied by their names, while JC (Jump on Carry) and JNC (Jump on No Carry) test whether or not the carry flag is set. All four are two-byte instructions, with the same format as Figure 15.c. JB (Jump on Bit), JNB (Jump on No Bit) and JBC (Jump on Bit then Clear Bit) can test any status bit or input pin with a three byte instruction; the second byte specifies which bit to test and the third gives the relative offset value.

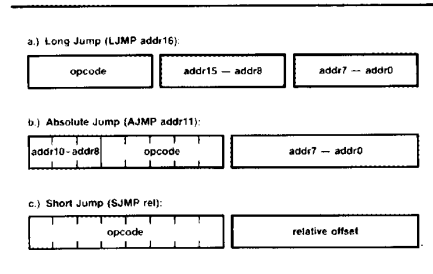
There are two subroutine-call instructions, LCALL (Long Call) and ACALL (Absolute Call). Each increments the P.C. to the first byte of the following instruction, then pushes it onto the stack (low byte first). Saving both bytes increments the stack pointer by two. The subroutine’s starting address is encoded in the same ways as LJMP and AJMP. The generic form of the call operation is the mnemonic CALL, which ASM51 will translate into LCALL or ACALL as appropriate.

The return instruction RET pops the high- and low-order bytes of the program counter successively from the stack, decrementing the stack pointer by two. Program execution continues at the address previously pushed; the first byte of the instruction immediately following the call.

When an interrupt request is recognized by the 8051 hardware, two things happen. Program control is automatically “vectored” to one of the interrupt service routine starting addresses by, in effect, forcing the CPU to process an LCALL instead of the next instruction. This automatically stores the return address on the stack. (Unlike the 8048, no status information is automatically saved.)

Secondly, the interrupt logic is disabled from accepting any other interrupts from the same or lower priority. After completing the interrupt service routine, executing an RETI (Return from Interrupt) instruction will return execution to the point where the background program was interrupted — just like RET — while restoring the interrupt logic to its previous state.

AFN-01502A-23



**Figure 15. Jump Instruction Machine Code Formats**



### Operate-and-branch instructions — CJNE, DJNZ

Two groups of instructions combine a byte operation with a conditional jump based on the results.

CJNE (Compare and Jump if Not Equal) compares two byte operands and executes a jump if they disagree. The carry flag is set following the rules for subtraction: if the unsigned integer value of the first operand is less than that of the second it is set; otherwise, it is cleared. However, neither operand is modified.

The CJNE instruction provides, in effect, a one-instruction "case" statement. This instruction may be executed repeatedly, comparing the code variable to a list of "special case" value: the code segment following the instruction (up to the destination label) will be executed only if the operands match. Comparing the accumulator or a register to a series of constants is a convenient way to check for special handling or error conditions; if none of the cases match the program will continue with "normal" processing.

A typical example might be a word processing device which receives ASCII characters through the serial port and drives a thermal hard-copy printer. A standard routine translates "printing" characters to bit patterns, but control characters (<DEL> <CR> <LF> <BEL> <ESC> or <SP>) must invoke corresponding special routines. Any other character with an ASCII code less than 20H should be translated into the <NUL> value, 00H, and processed with the printing characters.

Example 16 — Case Statements Using CJNE

```
CHAR EQU R7 ; CHARACTER CODE VARIABLE
INTP EQU CHAR, #7FH, INTP_1 ; (SPECIAL ROUTINE FOR RUBOUT CODE)
INTP_1 EQU CHAR, #07H, INTP_2 ; (SPECIAL ROUTINE FOR BELL CODE)
INTP_2 EQU CHAR, #0AH, INTP_3 ; (SPECIAL ROUTINE FOR LFEEED CODE)
INTP_3 EQU CHAR, #0DH, INTP_4 ; (SPECIAL ROUTINE FOR RETURN CODE)
INTP_4 EQU CHAR, #1BH, INTP_5 ; (SPECIAL ROUTINE FOR ESCAPE CODE)
INTP_5 EQU CHAR, #20H, INTP_6 ; (SPECIAL ROUTINE FOR SPACE CODE)
INTP_6 EQU JC PRINTC ; JUMP IF CODE > 20H
CHAR, #0 ; REPLACE CONTROL CHARACTERS WITH
PRINTC ; NULL CODE
; PROCESS STANDARD PRINTING
; CHARACTER
RET
```

DJNZ (Decrement and Jump if Not Zero) decrements the register or direct address indicated and jumps if the result is not zero, without affecting any flags. This provides a simple means for executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. For example, a 99-usec. software delay loop can be added to code forcing an I/O pin low with only two instructions.

Example 17 — Inserting a Software Delay with DJNZ

```
CLR R2, #49
DJNZ R2, $
SETB R2
```

The dollar sign in this example is a special character meaning "the address of this instruction." It is useful in eliminating instruction labels on the same or adjacent source lines. CJNE and DJNZ (like all conditional jumps) use program-counter relative addressing for the destination address.

### Stack Operations — PUSH, POP

The PUSH instruction increments the stack pointer by one, then transfers the contents of the single byte variable indicated (direct addressing only) into the internal RAM location addressed by the stack pointer. Conversely, POP copies the contents of the internal RAM location addressed by the stack pointer to the byte variable indicated, then decrements the stack pointer by one.

(Stack Addressing follows the same rules, and addresses the same locations as Register-indirect. Future microcomputers based on the MCS-51™ CPU could have up to 256 bytes of RAM for the stack.)

Interrupt service routines must not change any variable or hardware registers modified by the main program, or else the program may not resume correctly. (Such a change might look like a spontaneous random error.) Resources used or altered by the service routine (Accumulator, PSW, etc.) must be saved and restored to their previous value before returning from the service routine. PUSH and POP provide an efficient and convenient way to save register states on the stack.

Example 18 — Use of the Stack for Status Saving on Interrupts

```
LOC_THR EQU * ; REMEMBER LOCATION COUNTER
ORG 0003H ; STARTING ADDRESS FOR INTERRUPT ROUTINE
SERVER LUMP ; JUMP TO ACTUAL SERVICE ROUTINE LOCATED ELSEWHERE
;
ORG LOC_THR ; RESTORE LOCATION COUNTER
SERVER PUSH PSW ; SAVE ACCUMULATOR (NOTE DIRECT ADDRESSING
; NOTATION)
PUSH B ; SAVE B REGISTER
PUSH DPL ; SAVE DATA POINTER
PUSH DPH ;
MOV PSW, #00001000B ; SELECT REGISTER BANK 1
;
POP DPH ; RESTORE REGISTERS IN REVERSE ORDER
POP DPL ;
POP B ;
POP ACC ;
POP PSW ; RESTORE PSW AND RE-SELECT ORIGINAL
; REGISTER BANK
; RETURN TO MAIN PROGRAM AND RESTORE
; INTERRUPT LOGIC
RETI
```

If the SP register held 1FH when the interrupt was detected, then while the service routine was in progress the stack would hold the registers shown in Figure 16; SP would contain 26H.

The example shows the most general situation; if the service routine doesn't alter the B-register and data pointer, for example, the instructions saving and restoring those registers would not be necessary.

The stack may also pass parameters to and from subroutines. The subroutine can indirectly address the parameters derived from the contents of the stack pointer.

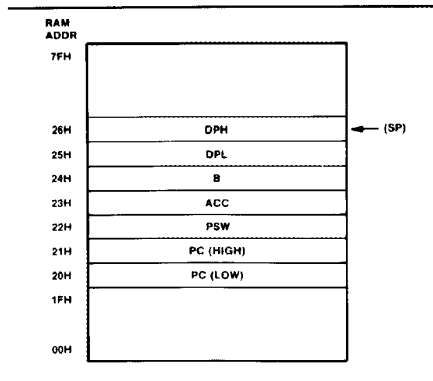


Figure 16. Stack contents during interrupt

One advantage here is simplicity. Variables need not be allocated for specific parameters, a potentially large number of parameters may be passed, and different calling programs may use different techniques for determining or handling the variables.

For example, the following subroutine reads out a parameter stored on the stack by the calling program, uses the low order bits to access a local look-up table holding bit patterns for driving the coils of a four phase stepper motor, and stores the appropriate bit pattern back in the same position on the stack before returning. The accumulator contents are left unchanged.

Example 19 — Passing Variable Parameters to Subroutines Using the Stack

```

NXTPOS MOV R0, SP      ; ACCESS LOCATION PARAMETER PUSHED INTO
DEC R0      ; ACCESS LOCATION PARAMETER PUSHED INTO
DEC R0      ; ACCESS LOCATION PARAMETER PUSHED INTO
XCH A, R0    ; READ INPUT PARAMETER AND SAVE
            ; ACCUMULATOR
ANL A, #03H  ; MASK ALL BUT LOW-ORDER TWO BITS
ADD A, #2    ; ALLOW FOR OFFSET FROM MOVX TO TABLE
MOVC A, @A+PC ; READ LOOK-UP TABLE ENTRY
XCH A, R0    ; PASS BACK TRANSLATED VALUE AND RESTORE
            ; ACC
RET          ; RETURN TO BACKGROUND PROGRAM
STPTBL DB 01011111B ; POSITION 0
        DB 01011111B ; POSITION 1
        DB 10011111B ; POSITION 2
        DB 10101111B ; POSITION 3
    
```

The background program may reach this subroutine with several different calling sequences, all of which PUSH a value before calling the routine and POP the result after. A motor on Port 1 may be initialized by placing the desired position (zero) on the stack before calling the subroutine and outputting the results directly to a port afterwards.

Example 20 — Sending and Receiving Data Parameters Via the Stack

```

CLR A
PUSH ACC
CALL NXTPOS
POP P1
    
```

If the position of the motor is determined by the contents of variable POSM1 (a byte in internal RAM) and the position of a second motor on Port 2 is determined by the data input to the low-order nibble of Port 2, a six-instruction sequence could update them both.

Example 21 — Loading and Unloading Stack Direct from I/O Ports

```

POSM1 EQU S1
PUSH POSM1
CALL NXTPOS
POP P1
PUSH P2
CALL NXTPOS
POP P2
    
```

## Data Pointer and Table Look-up instructions — MOV, INC, MOVC, JMP

The data pointer can be loaded with a 16-bit value using the instruction MOV DPTR, #data16. The data used is stored in the second and third instruction bytes, high-order byte first. The data pointer is incremented by INC DPTR. A 16-bit increment is performed; an overflow from the low byte will carry into the high-order byte. Neither instruction affects any flags.

The MOVC (Move Constant) instructions (MOVC A, @A+DPTR and MOVC A, @A+PC) read into the accumulator bytes of data from the program memory logical address space. Both use a form of indexed addressing; the former adds the unsigned eight-bit accumulator contents with the sixteen-bit data pointer register, and uses the resulting sum as the address from which the byte is fetched. A sixteen-bit addition is performed; a carry-out from the low-order eight bits may propagate through higher-order bits, but the contents of the DPTR are not altered. The latter form uses the incremented program counter as the “base” value instead of the DPTR (figure 17). Again, neither version affects the flags.

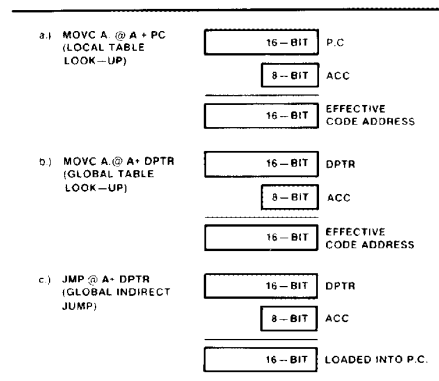


Figure 17. Operation of MOVC instructions

AFN-01502A-25

Each can be part of a three step sequence to access look-up tables in ROM. To use the DPTR-relative version, load the Data Pointer with the starting address of a look-up table; load the accumulator with (or compute) the index of the entry desired; and execute `MOVC A,@A+DPTR`. Unlike the similar `MOVP3` instructions in the 8048, the table may be located anywhere in program memory. The data pointer may be loaded with a constant for short tables. Or to allow more complicated data structures, or tables with more than 256 entries, the values for `DPH` and `DPL` may be computed or modified with the standard arithmetic instruction set.

The PC-relative version has the advantage of not affecting the data pointer. Again, a look-up sequence takes three steps: load the accumulator with the index; compensate for the offset from the look-up instruction to the start of the table by adding the number of bytes separating them to the accumulator; then execute the `MOVC A,@A+PC` instruction.

Let's look at a non-trivial situation where this instruction would be used. Some applications store large multi-dimensional look-up tables of dot matrix patterns, non-linear calibration parameters, and so on in a linear (one-dimensional) vector in program memory. To retrieve data from the tables, variables representing matrix indices must be converted to the desired entry's memory address. For a matrix of dimensions (`MDIMEN` x `NDIMEN`) starting at address `BASE` and respective indices `INDEXI` and `INDEXJ`, the address of element (`INDEXI`, `INDEXJ`) is determined by the formula.

$$\text{Entry Address} = \text{BASE} + (\text{NDIMEN} \times \text{INDEXI}) + \text{INDEXJ}$$

The code shown below can access any array with less than 255 entries (i.e., an `11x21` array with 231 elements). The table entries are defined using the Data Byte ("DB") directive, and will be contained in the assembly object code as part of the accessing subroutine itself.

#### Example 22—Use of MPY and Data Pointer Instructions to Access Entries from a Multi-dimensional Look-Up Table in ROM

```

;MATRIX1 LOAD CONSTANT READ FROM TWO DIMENSIONAL LOOK-UP
;
; TABLE IN PROGRAM MEMORY INTO ACCUMULATOR
;
; USING LOCAL TABLE LOOK-UP INSTRUCTION, "MOVC A,@A+PC"
; THE TOTAL NUMBER OF TABLE ENTRIES IS ASSUMED TO
; BE SMALL, (E LESS THAN ABOUT 256 ENTRIES)
;
; TABLE USED IN THIS EXAMPLE IS ( 11 X 21 )
; DESIRED ENTRY ADDRESS IS GIVEN BY THE FORMULA:
; ( (BASE ADDRESS) + (21 X INDEXI) + (INDEXJ) )
;
INDEXI EQU R6 ; FIRST COORDINATE OF ENTRY (0-10)
INDEXJ EQU R7H ; SECOND COORDINATE OF ENTRY (0-20)
;
MATRIX1 MOV A,INDEXI
MOV R,#21
MUL AB
ADD A,INDEXJ
; ALLOW FOR INSTRUCTION BYTE BETWEEN "MOVC" AND
; ENTRY (0,0)
INC A
MOVC A,@A+PC
RET

BASE1 DB 1 ; (entry 0,0)
DB 2 ; (entry 0,1)
;
DB 21 ; (entry 0,20)
DB 22 ; (entry 1,0)
;
DB 42 ; (entry 1,20)
;
DB 231 ; (entry 10,20)

```

There are several different means for branching to sections of code determined or selected at run time. (The single destination addresses incorporated into conditional and unconditional jumps are, of course, determined at assembly time). Each has advantages for different applications.

The most common is an N-way conditional jump based on some variable, with all of the potential destinations known at assembly time. One of a number of small routines is selected according to the value of an index variable determined while the program is running. The most efficient way to solve this problem is with the `MOVC` and an indirect jump instruction, using a short table of one byte offset values in ROM to indicate the relative starting addresses of the several routines.

`JMP @A+DPTR` is an instruction which performs an indirect jump to an address determined during program execution. The instruction adds the eight-bit unsigned accumulator contents with the contents of the sixteen-bit data pointer, just like `MOVC A,@A+DPTR`. The resulting sum is loaded into the program counter and is used as the address for subsequent instruction fetches. Again, a sixteen-bit addition is performed; a carry out from the low-order eight bits may propagate through the higher-order bits. In this case, neither the accumulator contents nor the data pointer is altered.

The example subroutine below reads a byte of RAM into the accumulator from one of four alternate address spaces, as selected by the contents of the variable `MEMSEL`. The address of the byte to be read is determined by the contents of `R0` (and optionally `R1`). It might find use in a printing terminal application, where four different model printers all use the same ROM code but use different types and sizes of buffer memory for different speeds and options.

#### Example 23—N-Way Branch and Computed Jump Instructions via `JMP @ADPTR`

```

MEMSEL EQU R3
JUMP_4 MOV A, MEMSEL
MOV DPTR, JUMPTBL
MOVC A,@A+DPTR
JMP @A+DPTR
JUMPTBL DB MEMSP0-JUMPTBL
DB MEMSP1-JUMPTBL
DB MEMSP2-JUMPTBL
DB MEMSP3-JUMPTBL
MEMSP0 MOV A,#00 ; READ FROM INTERNAL RAM
RET
MEMSP1 MOVX A,#00 ; READ FROM 256 BYTES OF EXTERNAL RAM
RET
MEMSP2 DPL,R0
MOV DPH,R1
MOVX A,@DPTR ; READ FROM 64K BYTES OF EXTERNAL RAM
RET
MEMSP3 MOV A,R1
ANL A,#07H
ANL P1,#1111000B
ORL P1,A
MOVX A,#00 ; READ FROM 4K BYTES OF EXTERNAL RAM
RET

```

Note that this approach is suitable whenever the size of jump table plus the length of the alternate routines is less than 256 bytes. The jump table and routines may be located anywhere in program memory, independent of 256-byte program memory pages.

AFN-01502A-26

For applications where up to 128 destinations must be selected, all of which reside in the same 2K page of program memory which may be reached by the two-byte absolute jump instructions, the following technique may be used. In the above mentioned printing terminal example, this sequence could "parse" 128 different codes for ASCII characters arriving via the 8051 serial port.

Example 24—N-Way Branch with 128 Optional Destinations

```
OPTION EQU R3
;
JMP128 MOV A,OPTION ;MULTIPLY BY 2 FOR 2 BYTE JUMP TABLE
RL A ;
MOV DPTR,#INSTBL ;FIRST ENTRY IN JUMP TABLE
JMP @A+DPTR ;JUMP INTO JUMP TABLE
;
INSTBL AJMP PROC00 ;128 CONSECUTIVE
AJMP PROC01 ;JUMP INSTRUCTIONS
AJMP PROC02
;
AJMP PROC7E
AJMP PROC7F
```

The destinations in the jump table (PROC00-PROC7F) are not all necessarily unique routines. A large number of special control codes could each be processed with their own unique routine, with the remaining printing characters all causing a branch to a common routine for entering the character into the output queue. In those rare situations where even 128 options are insufficient, or where the destination routines may cross a 2K page boundary, the above approach may be modified slightly as shown below.

Example 25—256-Way Branch Using Address Look-Up Tables

```
RTEMP EQU R7
;
JMP256 MOV DPTR,#ADRTBL ;FIRST ENTRY IN TABLE OF ADDRESSED
MOV A,OPTION
CLR C
RLC A ;MULTIPLY BY 2 FOR 2 BYTE JUMP TABLE
JNC LON128
JNC DPH
;
LON128 MOV RTEMP,A ;SAVE ACC FOR HIGH BYTE READ
XCH A,A+DPTR ;READ LOW BYTE FROM JUMP TABLE
INCL A
;
MOV C,A+DPTR ;GET LOW-ORDER BYTE FROM TABLE
PUSH ACC
MOV A,RTEMP
MOV C,A+DPTR ;GET HIGH-ORDER BYTE FROM TABLE
PUSH ACC
;
; THE TWO ACC PUSHES HAVE PRODUCED
; A "RETURN ADDRESS" ON THE STACK WHICH CORRESPONDS
; TO THE DESIRED STARTING ADDRESS
; IT MAY BE REACHED BY POPPING THE STACK
; INTO THE PC
RET
;
ADRTBL DW PROC00 ;UP TO 256 CONSECUTIVE DATA
DW PROC01 ;WORDS INDICATING STARTING ADDRESSES
;
DW PROC7F
;
; DUMMY CODE ADDRESS DEFINITIONS NEEDED BY ABOVE
; TWO EXAMPLES
;
PROC00 NOP
PROC01 NOP
PROC02 NOP
PROC7E NOP
PROC7F NOP
PROC7F NOP
```

#### 4. BOOLEAN PROCESSING INSTRUCTIONS

The commonly accepted terms for tasks at either end of the computational vs. control application spectrum are, respectively, "number-crunching" and "bit-banging".

Prior to the introduction of the MCS-51™ family, nice number-crunchers made bad bit-bangers and vice versa. The 8051 is the industry's first single-chip micro-computer designed to crunch **and** bang. (In some circles, the latter technique is also referred to as "bit-twiddling". Either is correct.)

#### Direct Bit Addressing

A number of instructions operate on Boolean (one-bit) variables, using a direct bit addressing mode comparable to direct byte addressing. An additional byte appended to the opcode specifies the Boolean variable, I/O pin, or control bit used. The state of any of these bits may be tested for "true" or "false" with the conditional branch instructions JB (Jump on Bit) and JNB (Jump on Not Bit). The JBC (Jump on Bit and Clear) instruction combines a test-for-true with an unconditional clear.

As in direct byte addressing, bit 7 of the address byte switches between two physical address spaces. Values between 0 and 127 (00H-7FH) define bits in internal RAM locations 20H to 2FH (Figure 18a); address bytes between 128 and 255 (80H-0FFH) define bits in the 2 x "special-function" register address space (Figure 18b). If no 2 x "special-function" register corresponds to the direct bit address used the result of the instruction is undefined.

Bits so addressed have many wondrous properties. They may be set, cleared, or complemented with the two byte instructions SETB, CLR, or CPL. Bits may be moved to and from the carry flag with MOV. The logical ANL and ORL functions may be performed between the carry and either the addressed bit or its complement.

#### Bit Manipulation Instructions — MOV

The "MOV" mnemonic can be used to load an addressable bit into the carry flag ("MOV C, bit") or to copy the state of the carry to such a bit ("MOV bit, C"). These instructions are often used for implementing serial I/O algorithms via software or to adapt the standard I/O port structure.

It is sometimes desirable to "re-arrange" the order of I/O pins because of considerations in laying out printed circuit boards. When interfacing the 8051 to an immediately adjacent device with "weighted" input pins, such as keyboard column decoder, the corresponding pins are likely to be not aligned (Figure 19).

There is a trade-off in "scrambling" the interconnections with either interwoven circuit board traces or through software. This is extremely cumbersome (if not impossible) to do with byte-oriented computer architectures. The 8051's unique set of Boolean instructions makes it simple to move individual bits between arbitrary locations.

AFN-01502A-27

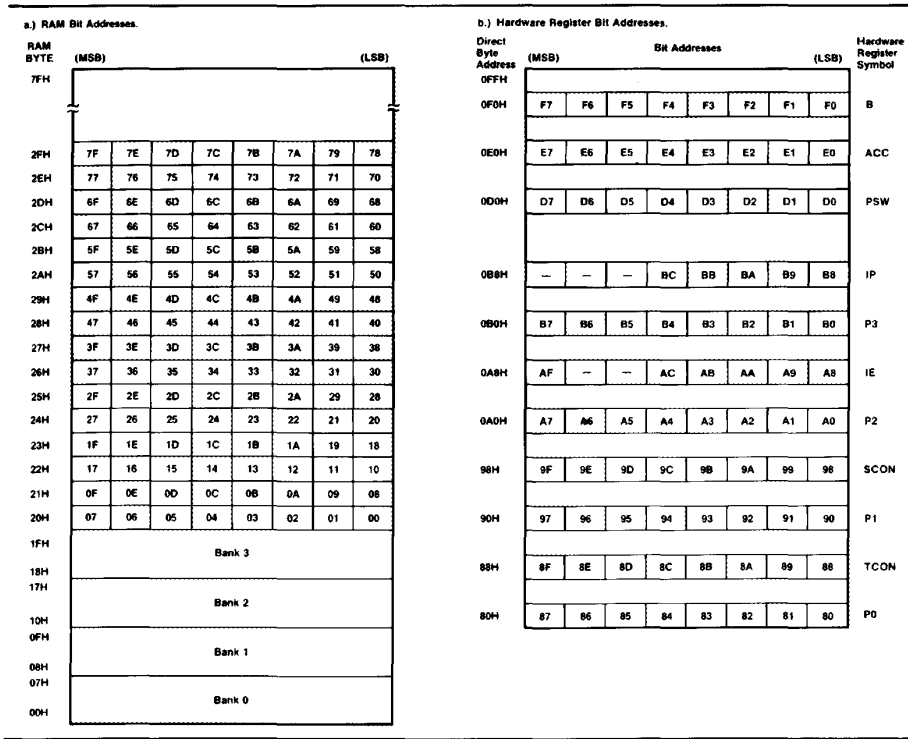
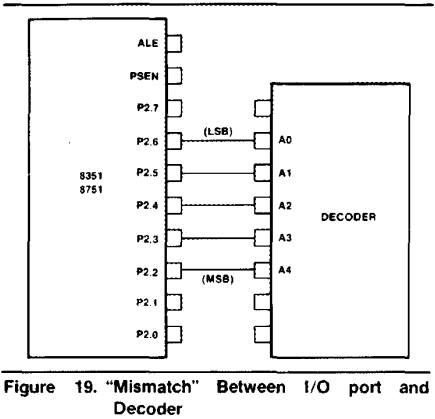


Figure 18. Bit Address Maps



Example 26—Re-ordering I/O Port Configuration

```
OUT_P2 RRC A ; MOVE ORIGINAL ACC 0 INTO CY
      MOV P2 8;C ; STORE CARRY TO PIN P28
      RRC A ; MOVE ORIGINAL ACC 1 INTO CY
      MOV P2 5;C ; STORE CARRY TO PIN P25
      RRC A ; MOVE ORIGINAL ACC 2 INTO CY
      MOV P2 4;C ; STORE CARRY TO PIN P24
      RRC A ; MOVE ORIGINAL ACC 3 INTO CY
      MOV P2 3;C ; STORE CARRY TO PIN P23
      RRC A ; MOVE ORIGINAL ACC 4 INTO CY
      MOV P2 2;C ; STORE CARRY TO PIN P22
      RET
```

Solving Combinatorial Logic Equations — ANL, ORL

Virtually all hardware designers are familiar with the problem of solving complex functions using combinatorial logic. The technologies involved may vary greatly, from multiple contact relay logic, vacuum tubes, TTL, or CMOS to more esoteric approaches like fluidics, but in each case the goal is the same: a Boolean (true/false) function is computed on a number of Boolean variables.

APN-01502A-28

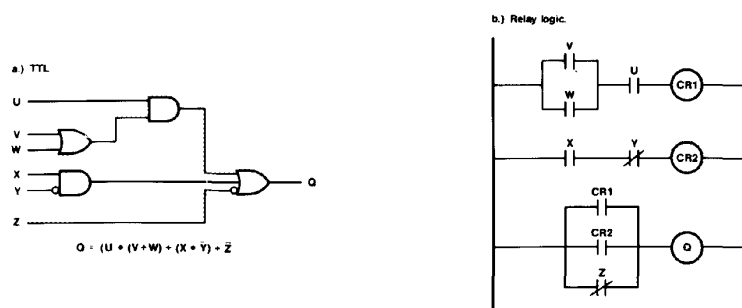


Figure 20. Implementations of Boolean functions

Figure 20 shows the logic diagram for an arbitrary function of six variables named U through Z using standard logic and relay logic symbols. Each is a solution of the equation,

$$Q = (U * (V + W) + (X * \bar{Y}) + \bar{Z}$$

(While this equation could be reduced using Karnaugh Maps or algebraic techniques, that is not the purpose of this example. Even a minor change to the function equation would require re-reducing from scratch.)

Most digital computers can solve equations of this type with standard word-wide logical instructions and conditional jumps. Still, such software solutions seem somewhat sloppy because of the many paths through the program the computation can take.

Assume U and V are input pins being read by different input ports. W and X are status bits for two peripheral controllers (read as I/O ports), and Y and Z are software flags set or cleared earlier in the program. The end result must be written to an output pin on some third port.

For the sake of comparison we will implement this function with software drawn from three proper subsets of the MCS-51™ instruction set. The first two implementations follow the flow chart shown in Figure 21. Program flow would embark on a route down a test-and-branch tree and leaves either the "True" or "Not True" exit ASAP. These exits then write the output port with the data previously written to the same port with the result bit respectively one or zero.

In the first case, we assume there are no instructions for addressing individual bits other than special flags like the carry. This is typical of many older microprocessors and mainframe computers designed for number-crunching. MCS-51™ mnemonics are used here, though for most other machines the issue would be even further clouded by their use of operation-specific mnemonics like

INPUT, OUTPUT, LOAD, STORE, etc., instead of the universal MOV.

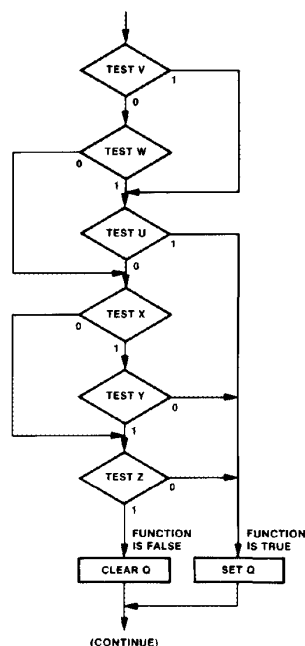


Figure 21. Flow chart for tree-branching logic implementation

AFN-01502A-29

Example 27—Software Solution to Logic Function of Figure 20, Using only Byte-Wide Logical Instructions

```

;BFUNC1 SOLVE A RANDOM LOGIC FUNCTION OF 6
;VARIABLES BY LOADING AND MASKING THE APPROPRIATE
;BITS IN THE ACCUMULATOR, THEN EXECUTING CONDITIONAL
;JUMPS BASED ON ZERO CONDITION
; (APPROACH USED BY BYTE-ORIENTED ARCHITECTURES )
; BYTE AND MASK VALUES CORRESPOND TO RESPECTIVE
; BYTE ADDRESS AND BIT POSITION
OUTBUF EQU 22H ; OUTPUT PIN STATE MAP
TESTV MOV A,P2
ANL A,#00000100B
JNZ TESTU
MOV A,TC0N
ANL A,#00100000B
JZ TESTX
MOV A,P1
ANL A,#00000001B
JNZ SET0
MOV A,TC0N
ANL A,#00001000B
JZ TESTZ
MOV A,20H
ANL A,#00000001B
JZ SET0
TESTZ MOV A,21H
ANL A,#00000010B
JZ SET0
MOV A,OUTBUF
CLRQ ANL A,#11110111B
JMP OUT0
MOV A,OUTBUF
SET0 ORL A,#00001000B
OUT0 MOV OUTBUF,A
MOV P3,A

```

Cumbersome, to say the least, and error prone. It would be hard to prove the above example worked in all cases without an exhaustive test.

Each move/mask/conditional jump instruction sequence may be replaced by a single bit-test instruction thanks to direct bit addressing. But the algorithm would be equally convoluted.

Example 28—Software Solution to Logic Function of Figure 20, Using only Bit-Test Instructions

```

;BFUNC2 SOLVE A RANDOM LOGIC FUNCTION OF 6
;VARIABLES BY DIRECTLY POLLING EACH BIT
; (APPROACH USING MCS-51 UNIQUE BIT-TEST
; INSTRUCTION CAPABILITY )
; SYMBOLS USED IN LOGIC DIAGRAM ASSIGNED TO
; CORRESPONDING 8051 BIT ADDRESSES
;
U BIT P1.1
V BIT P2.2
W BIT TFO
X BIT IC1
Y BIT 20H.0
Z BIT 21H.1
Q BIT P3.3
;
TEST_V JB V,TEST_U
TEST_U JNB W,TEST_X
TEST_X JNB X,TEST_Z
TEST_Z JNB Y,SET_Q
CLR_Q CLR Q
SET_Q JMP NXTTST
NXTTST SETB Q
; (CONTINUATION OF PROGRAM)

```

A more elegant and efficient 8051 implementation uses the Boolean ANL and ORL functions to generate the output function using straight-line code. These instructions perform the corresponding logical operations between the carry flag ("Boolean Accumulator") and the addressed bit, leaving the result in the carry. Alternate forms of each instruction (specified in the assembly language by placing a slash before the bit name) use the complement of the bit's state as the input operand.

These instructions may be "strung together" to simulate a multiple input logic gate. When finished, the carry flag contains the result, which may be moved directly to the destination or output pin. No flow chart is needed—it is simple to code directly from the logic diagrams in Figure 20.

Example 29—Software Solution to Logic Function of Figure 20, Using the MCS-51 (TM) Unique Logical Instructions on Boolean Variables

```

;BFUNC3 SOLVE A RANDOM LOGIC FUNCTION OF 6
;VARIABLES USING STRAIGHT-LINE LOGICAL INSTRUCTIONS
; ON MCS-51 BOOLEAN VARIABLES
;
MOV C,V ; OUTPUT OF OR GATE
ORL C,W ; OUTPUT OF TOP AND GATE
MOV FO,C ; SAVE INTERMEDIATE STATE
MOV C,X
ANL C,Y ; OUTPUT OF BOTTOM AND GATE
ORL C,FO ; INCLUDE VALUE SAVED ABOVE
ORL C,Z ; INCLUDE LAST INPUT VARIABLE
MOV G,C ; OUTPUT COMPUTED RESULT

```

Simplicity itself. Fast, flexible, reliable, easy to design, and easy to debug.

The Boolean features are useful and unique enough to warrant a complete Application Note of their own. Additional uses and ideas are presented in Application Note AP-70, *Using the Intel® MCS-51® Boolean Processing Capabilities*, publication number 121519.

## 5. ON-CHIP PERIPHERAL FUNCTION OPERATION AND INTERFACING

### I/O Ports

The I/O port versatility results from the "quasi-bidirectional" output structure depicted in Figure 22. (This is effectively the structure of ports 1, 2, and 3 for normal I/O operations. On port 0 resistor R2 is disabled except during multiplexed bus operations, providing

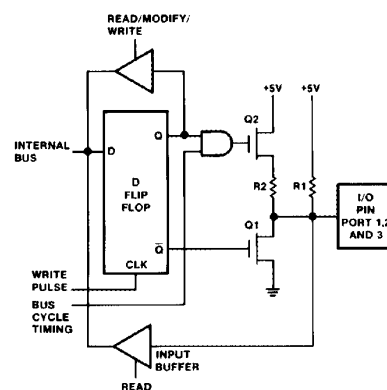


Figure 22. Pseudo-bidirectional I/O port circuitry

AFN-01502A-30

essentially open-collector outputs. For full electrical characteristics see the User's Manual.)

An output latch bit associated with each pin is updated by direct addressing instructions when that port is the destination. The latch state is buffered to the outside world by R1 and Q1, which may drive a standard TTL input. (In TTL terms, Q1 and R1 resemble an open-collector output with a pull-up resistor to Vcc.)

R2 and Q2 represent an "active pull-up" device enabled momentarily when a 0 previously output changes to a 1. This "jerks" the output pin to a 1 level more quickly than the passive pull-up, improving rise-time significantly if the pin is driving a capacitive load. Note that the active pull-up is **only** activated on 0-to-1 transitions at the output latch (unlike the 8048, in which Q2 is activated whenever a 1 is written out).

Operations using an input port or pin as the source operand use the logic level of the pin itself, rather than the output latch contents. This level is affected by both the microcomputer itself and whatever device the pin is connected to externally. The value read is essentially the "OR-tied" function of Q1 and the external device. If the external device is high-impedance, such as a logic gate input or a three state output in the third state, then reading a pin will reflect the logic level previously output. To use a pin for input, the corresponding output latch must be set. The external device may then drive the pin with either a high or low logic signal. Thus the same port may be used as both input and output by writing ones to all pins used as inputs on output operations, and ignoring all pins used as output on an input operation.

In one operand instructions (INC, DEC, DJNZ and the Boolean CPL) the output latch rather than the input pin level is used as the source data. Similarly, two operand instructions using the port as both one source and the destination (ANI, ORI, XRI) use the output latches. This ensures that latch bits corresponding to pins used as inputs will not be cleared in the process of executing these instructions.

The Boolean operation JBC tests the output latch bit, rather than the input pin, in deciding whether or not to jump. Like the byte-wise logical operations, Boolean operations which modify individual pins of a port leave the other bits of the output latch unchanged.

A good example of how these modes may play together may be taken from the host-processor interface expected by an 8243 I/O expander. Even though the 8051 does not include 8048-type instructions for interfacing with an 8243, the parts can be interconnected (Figure 23) and the protocol may be emulated with simple software.

### Example 30 — Mixing Parallel Output, Input, and Control Strokes on Port 2

```

IN0243 INPUT DATA FROM AN 8243 I/O EXPANDER
CONNECTED TO P23-P20
PORT 1 FROM MICRO CS = PROG
P01-P06 USED AS INPUTS
PORT 2 TO BE READ IN ACC

IN0243 OR A,#11100000
MOV P0,A OUTPUT INSTRUCTION CODE
CLR P04 FALLING EDGE OF PACK
CPL P04
CPL P04
MOV A,P2 READ INPUT DATA
SETC P04 RETURN FLAG HIGH
SETC P05 DE-SELECT CHIP

```

### Serial Port and Timer applications

Configuring the 8051's Serial Port for a given data rate and protocol requires essentially three short sections of software. On power-up or hardware reset the serial port and timer control words must be initialized to the appropriate values. Additional software is also needed in the transmit routine to load the serial port data register and in the receive routine to unload the data as it arrives.

This is best illustrated through an arbitrary example. Assume the 8051 will communicate with a CRT operating at 2400 baud (bits per second). Each character is transmitted as seven data bits, odd parity, and one stop bit. This results in a character rate of 2400 / 10 = 240 characters per second.

For the sake of clarity, the transmit and receive subroutines are driven by simple-minded software status polling code rather than interrupts. (It might help to refer back to Figures 7-9 showing the control word formats.) The serial port must be initialized to 8-bit UART mode (M0, M1=01), enabled to receive all messages (M2=0, REN=1). The flag indicating that the transmit register is free for more data will be artificially set in order to let the output software know the output register is available. This can all be set up with one instruction.

### Example 31 — Serial Port Mode and Control Bits

```

SPINIT INITIALIZE SERIAL PORT
FOR 8-BIT UART MODE
& SET TRANSMIT READY FLAG
SPINIT MOV SCON,#01010010B

```

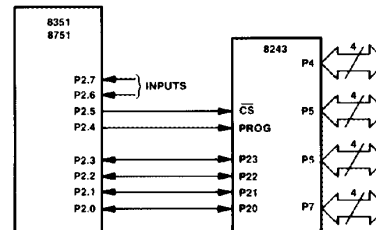


Figure 23. Connecting an 8051 with an 8243 I/O Expander

AFN-01502A-31



Timer 1 will be used in auto-reload mode as a data rate generator. To achieve a data rate of 2400 baud, the timer must divide the 1 MHz internal clock by 32 x (desired data rate):

$$\frac{1 \times 10^6}{(32)(2400)}$$

which equals 13.02 rounded down to 13 instruction cycles. The timer must reload the value -13, or 0F3H. (ASM51 will accept both the signed decimal or hexadecimal representations.)

#### Example 32—Initializing Timer Mode and Control Bits

```

;T1INIT INITIALIZE TIMER 1 FOR
;      AUTO-RELOAD AT 32*2400 HZ
;      (TO USED AS GATED 16-BIT COUNTER )
T1INIT  MOV     TCON, #11010010B
        MOV     TH1, #-13
        SETB    TR1

```

A simple subroutine to transmit the character passed to it in the accumulator must first compute the parity bit, insert it into the data byte, wait until the transmitter is available, output the character, and return. This is nearly as easy said as done.

#### Example 33—Code for UART Output, Adding Parity, Transmitter Loading

```

;SP_OUT ADD ODD PARITY TO ACC AND
;      TRANSMIT WHEN SERIAL PORT READY
SP_OUT  MOV     C, P
        CPL     C
        MOV     ACC, 7: C
        JNB     T1, $
        CLR     T1
        MOV     SBUF, A
        RET

```

A simple minded routine to wait until a character is received, set the carry flag if there is an odd-parity error, and return the masked seven-bit code in the accumulator is equally short.

#### Example 34—Code for UART Reception and Parity Verification

```

;SP_IN  INPUT NEXT CHARACTER FROM SERIAL PORT
;      SET CARRY IF ODD-PARITY ERROR
SP_IN   JNB     RI, $
        CLR     RI
        MOV     A, SBUF
        MOV     C, P
        CPL     C
        ANL     A, #7FH
        RET

```

## 6. SUMMARY

This Application Note has described the architecture, instruction set, and on-chip peripheral features of the first three members of the MCS-51™ microcomputer family. The examples used throughout were admittedly (and necessarily) very simple. Additional examples and techniques may be found in the MCS-51™ User's Manual and other application notes written for the MCS-48™ and MCS-51™ families.

Since its introduction in 1977, the MCS-48™ family has become the industry standard single-chip microcomputer. The MCS-51™ architecture expands the addressing capabilities and instruction set of its predecessor while ensuring flexibility for the future, and maintaining basic software compatibility with the past.

Designers already familiar with the 8048 or 8049 will be able to take with them the education and experience gained from past designs as ever-increasing system performance demands force them to move on to state-of-the-art products. Newcomers will find the power and regularity of the 8051 instruction set an advantage in streamlining both the learning and design processes.

Microcomputer system designers will appreciate the 8051 as basically a single-chip solution to many problems which previously required board-level computers. Designers of real-time control systems will find the high execution speed, on-chip peripherals, and interrupt capabilities vital in meeting the timing constraints of products previously requiring discrete logic designs. And designers of industrial controllers will be able to convert ladder diagrams directly from tested-and-true TTL or relay-logic designs to microcomputer software, thanks to the unique Boolean processing capabilities.

It has not been the intent of this note to gloss over the difficulty of designing microcomputer-based systems. To be sure, the hardware and software design aspects of any new computer system are nontrivial tasks. However, the system speed and level of integration of the MCS-51™ microcomputers, the power and flexibility of the instruction set, and the sophisticated assembler and other support products combine to give both the hardware and software designer as much of a head start on the problem as possible.

***EX PARTE REEXAMINATION***

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

|                                |  |   |                 |                 |
|--------------------------------|--|---|-----------------|-----------------|
| Reexamination<br>Control. No.: | 90/011,754   | § | Atty.Dkt.No.:   | 5707-26200      |
| Inventor:                      | Ian F. Harris, et al.  | § | Examiner:       | Peikari, Behzad |
| Request Filing Date:           | June 21, 2011  | § | Group/Art Unit: | 3992            |
| U.S. Patent No.:               | 7,930,576  | § | Conf. No.       | 4071            |
| Title:                         | SHARING NON-SHARABLE<br>DEVICES BETWEEN AN<br>EMBEDDED CONTROLLER<br>AND A PROCESSOR IN A<br>COMPUTER SYSTEM | § |                 |                 |

**\*\*\*\*CERTIFICATE OF E-FILING TRANSMISSION\*\*\*\***

I hereby certify that this correspondence is being transmitted  
via electronic filing to the United States Patent and  
Trademark Office on the date shown below:

On: April 23, 2012                      /Anthony M. Petro/  
Date    Anthony M. Petro, #59,391

**CERTIFICATE OF SERVICE**

The undersigned hereby certifies that a true and correct copy of the Response to Office action Mailed February 23, 2012 and Declaration of Thomas M. Conte Under 37 C.F.R. § 1.132, which were filed with the United States Patent & Trademark Office electronically via EFS-Web on April 23, 2012, has been deposited with the United States Postal Service and sent via First Class Mail on April 23, 2012, to the Third Party Requester at the following address:

Jiawei Huang  
J.C. Patents  
4 Venture, Suite 250  
Irvine, CA 92618

Respectfully submitted,

Date: April 23, 2012

By: /Anthony M. Petro/  
Anthony M. Petro  
Reg. No. 59,391

Meyertons, Hood, Kivlin, Kowert & Goetzl, P.C.  
P.O. Box 398  
Austin, TX 78767-0398  
(512) 853-8800

| Electronic Acknowledgement Receipt          |  |
|---|--|
| <b>EFS ID:</b>                              | 12603784   |
| <b>Application Number:</b>                  | 90011754   |
| <b>International Application Number:</b>    |  |
| <b>Confirmation Number:</b>                 | 4071   |
| <b>Title of Invention:</b>                  | SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM |
| <b>First Named Inventor/Applicant Name:</b> | 7,930,576 B2   |
| <b>Customer Number:</b>                     | 35690  |
| <b>Filer:</b>                               | Anthony M. Petro/Jeannie Miranda   |
| <b>Filer Authorized By:</b>                 | Anthony M. Petro   |
| <b>Attorney Docket Number:</b>              | JCLA38862-RE   |
| <b>Receipt Date:</b>                        | 23-APR-2012  |
| <b>Filing Date:</b>                         | 21-JUN-2011  |
| <b>Time Stamp:</b>                          | 14:21:09   |
| <b>Application Type:</b>                    | Reexam (Third Party)   |

### Payment information:

|                        |    |
|------------------------|----|
| Submitted with Payment | no |
|------------------------|----|

### File Listing:

| Document Number | Document Description | File Name                              | File Size(Bytes)/ Message Digest                   | Multi Part /.zip | Pages (if appl.) |
|-----------------|----------------------|--|--|------------------|------------------|
| 1               |                      | 5707-26200_Response_to_OA_02_23_12.pdf | 373405<br>c74dd22c0d8a97b6d6a27350d982a37ca5c48a6a | yes              | 72               |

|                              | Multipart Description/PDF files in .zip description   |                                       |   |     |    |
|------------------------------|---|---------------------------------------|---|-----|----|
|                              | Document Description                                  |                                       | Start   | End |    |
|                              | Amendment/Req. Reconsideration-After Non-Final Reject |                                       | 1   | 3   |    |
|                              | Claims  |                                       | 4   | 32  |    |
|                              | Applicant Arguments/Remarks Made in an Amendment      |                                       | 33  | 72  |    |
| Warnings:                    |   |                                       |   |     |    |
| Information:                 |   |                                       |   |     |    |
| 2                            | Rule 130, 131 or 132 Affidavits                       | Conte_declaration-FINAL-executed.pdf  | 125616<br>c27d5da2ba0c02a723e8dbe33d287bc7d01f1c0c  | no  | 21 |
| Warnings:                    |   |                                       |   |     |    |
| Information:                 |   |                                       |   |     |    |
| 3                            | Rule 130, 131 or 132 Affidavits                       | Exhibit_1.pdf                         | 7678351<br>4809cb2f08296d40c572d687e26f3dff806e3680 | no  | 8  |
| Warnings:                    |   |                                       |   |     |    |
| Information:                 |   |                                       |   |     |    |
| 4                            | Rule 130, 131 or 132 Affidavits                       | Exhibit_2.pdf                         | 2120564<br>39e9886ff4465413deb595de8ebbd1ba5fd0b15  | no  | 9  |
| Warnings:                    |   |                                       |   |     |    |
| Information:                 |   |                                       |   |     |    |
| 5                            | Rule 130, 131 or 132 Affidavits                       | Exhibit_3.pdf                         | 4718441<br>ea6a32540fae9c845fafdcb035498a5617419a9  | no  | 31 |
| Warnings:                    |   |                                       |   |     |    |
| Information:                 |   |                                       |   |     |    |
| 6                            | Reexam Certificate of Service                         | 5707-26200_Certificate_of_service.pdf | 77002<br>206f3d2c665ff0cd8d486ab59e8e233379f17e68   | no  | 1  |
| Warnings:                    |   |                                       |   |     |    |
| Information:                 |   |                                       |   |     |    |
| Total Files Size (in bytes): |   |                                       | 15093379  |     |    |

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

**New Applications Under 35 U.S.C. 111**


If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

**National Stage of an International Application under 35 U.S.C. 371**

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

**New International Application Filed with the USPTO as a Receiving Office**

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

|   |                         |   |
|---|-------------------------|---|
| <b>Reexamination</b><br> | Application/Control No. | Applicant(s)/Patent Under Reexamination |
|   | 90/011,754              | 7,930,576 B2 ET AL.                     |
|   | Certificate Date        | Certificate Number                      |

|   |                         |                                       |   |
|---|-------------------------|---------------------------------------|---|
| Requester   | Correspondence Address: | <input type="checkbox"/> Patent Owner | <input checked="" type="checkbox"/> Third Party |
| <p>MEYERTONS, HOOD, KIVLIN, KOWERT &amp; GOETZEL, P.C.<br/> P.O. BOX 398<br/> AUSTIN, TX 78767-0398</p> |                         |                                       |   |

|   |                            |                   |
|---|----------------------------|-------------------|
| LITIGATION REVIEW <input checked="" type="checkbox"/> | BJP<br>(examiner initials) | 2/14/12<br>(date) |
| Case Name   |                            | Director Initials |
| None  |                            | <i>ML for IY</i>  |
|   |                            |                   |
|   |                            |                   |
|   |                            |                   |
|   |                            |                   |

| COPENING OFFICE PROCEEDINGS |        |
|-----------------------------|--------|
| TYPE OF PROCEEDING          | NUMBER |
| 1. None                     |        |
| 2.                          |        |
| 3.                          |        |
| 4.                          |        |



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| APPLICATION NO.  | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|--|-------------|----------------------|---------------------|------------------|
| 90/011,754   | 06/21/2011  | 7,930,576 B2         | JCLA38862-RE        | 4071             |
| 35690  | 7590        | 02/23/2012           | EXAMINER            |                  |
| MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.<br>P.O. BOX 398<br>AUSTIN, TX 78767-0398 |             |                      | ART UNIT            | PAPER NUMBER     |

DATE MAILED: 02/23/2012

Please find below and/or attached an Office communication concerning this application or proceeding.



UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents  
United States Patent and Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

**MAILED**

**FEB 23 2012**

**DO NOT USE IN PALM PRINTER**

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.

P.O. BOX 398

AUSTIN, TX 78767-0398

**CENTRAL REEXAMINATION UNIT**

**EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM**

REEXAMINATION CONTROL NO. 90/011,754.

PATENT NO. 7,930,576 B2 ET.

ART UNIT 3992.

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).



|  |                                   |  |
|--|-----------------------------------|--|
| <b>Office Action in Ex Parte Reexamination</b> | <b>Control No.</b><br>90/011,754  | <b>Patent Under Reexamination</b><br>7,930,576 B2 ET |
|  | <b>Examiner</b><br>BEHZAD PEIKARI | <b>Art Unit</b><br>3992                              |

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

- a ☒ Responsive to the communication(s) filed on 21 June 2011.      b ☐ This action is made FINAL.  
c ☒ A statement under 37 CFR 1.530 has not been received from the patent owner.

A shortened statutory period for response to this action is set to expire 2 month(s) from the mailing date of this letter. Failure to respond within the period for response will result in termination of the proceeding and issuance of an *ex parte* reexamination certificate in accordance with this action. 37 CFR 1.550(d). **EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.550(c).** If the period for response specified above is less than thirty (30) days, a response within the statutory minimum of thirty (30) days will be considered timely.

**Part I THE FOLLOWING ATTACHMENT(S) ARE PART OF THIS ACTION:**

1. ☐ Notice of References Cited by Examiner, PTO-892.      3. ☐ Interview Summary, PTO-474.  
2. ☐ Information Disclosure Statement, PTO/SB/08.      4. ☐ \_\_\_\_\_.

**Part II SUMMARY OF ACTION**

- 1a. ☒ Claims 1-23 are subject to reexamination.  
1b. ☐ Claims \_\_\_\_\_ are not subject to reexamination.  
2. ☐ Claims \_\_\_\_\_ have been canceled in the present reexamination proceeding.  
3. ☐ Claims \_\_\_\_\_ are patentable and/or confirmed.  
4. ☒ Claims 1-23 are rejected.  
5. ☐ Claims \_\_\_\_\_ are objected to.  
6. ☐ The drawings, filed on \_\_\_\_\_ are acceptable.  
7. ☐ The proposed drawing correction, filed on \_\_\_\_\_ has been (7a) ☐ approved (7b) ☐ disapproved.  
8. ☐ Acknowledgment is made of the priority claim under 35 U.S.C. § 119(a)-(d) or (f).  
    a) ☐ All b) ☐ Some\* c) ☐ None of the certified copies have  
    1 ☐ been received.  
    2 ☐ not been received.  
    3 ☐ been filed in Application No. \_\_\_\_\_.  
    4 ☐ been filed in reexamination Control No. \_\_\_\_\_.  
    5 ☐ been received by the International Bureau in PCT application No. \_\_\_\_\_.  
    \* See the attached detailed Office action for a list of the certified copies not received.  
9. ☐ Since the proceeding appears to be in condition for issuance of an *ex parte* reexamination certificate except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte* Quayle, 1935 C.D. 11, 453 O.G. 213.  
10. ☐ Other: \_\_\_\_\_

cc: Requester (if third party requester)

U.S. Patent and Trademark Office

PTOL-466 (Rev. 08-06)

Office Action in Ex Parte Reexamination

Part of Paper No. 20120214

## DETAILED ACTION

### *Reexamination*

1. This is an *ex parte* reexamination of U.S. Patent Number 7,930,576 (the '576 patent) requested by a third party requester. Claims 1-23 are subject to reexamination.

### *Prosecution Summary*

2. The following is a brief summary of the prosecution to date in this *ex parte* reexamination proceeding:
  - On June 21, 2011, a request for *ex parte* reexamination of claims 1-23 of the '576 patent was filed by a third party requestor.
  - On September 8, 2011, the USPTO mailed a decision granting *ex parte* reexamination and ordering the reexamination of claims 1-23.

### *References*

3. The references discussed herein are as follows:
  - U.S. Patent No. 6,161,162 to DeRoo et al. ("DeRoo").

***SNQs Raised in the Request***

4. The order mailed September 8, 2011 identified the following SNQs:
- **Issue 1:** A SNQ as to claims 1-2, 4-7 and 9-10 is raised by DeRoo.
  - **Issue 2:** A SNQ as to claims 3, 8 and 11-23 is raised by DeRoo.

***Claim Rejections – Relevant Statutes***

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

(e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) and the Intellectual Property and High Technology Technical Amendments Act of 2002 do not apply when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. Therefore, the prior art date of the reference is determined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

Art Unit: 3992

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

### *Claim Rejections – Detailed Explanation*

#### **Issue 1**

*A SNQ as to claims 1-2, 4-7 and 9-10 is raised by DeRoo.*

7. The request proposed a rejection of claims 1-2, 4-7 and 9-10 under 35 U.S.C. § 102(b) as being anticipated by DeRoo. However, DeRoo was not published until December 12, 2000, which was after the December 18, 2007 filing date of the '576 patent.

Thus, a rejection under 35 U.S.C. § 102(b) would be improper.

The following rejection under 35 U.S.C. § 102(e) is made *sua sponte*.

8. Claims 1-2, 4-7 and 9-10 are rejected under 35 U.S.C. § 102(e) as being anticipated by DeRoo.

With regard to claim 1, DeRoo discloses a system (*see FIG. 20*), comprising: a non-volatile memory (*FIG. 20, common memory device 704*); a processor (*FIG. 20, CPU 702*); and a microcontroller (*FIG. 20, SCP 706 and HUI 700*) coupled to the non-volatile memory and to the processor, wherein the microcontroller is configured to: in response to a change in system state

Art Unit: 3992

to a first state (*FIG. 24, when the CPU SYSTEM ACCESS signal is in low level*) wherein the microcontroller is assured safe access to the non-volatile memory, hold the system in the first state (*note address latch enable N\_SCPALE*) and switch access (*note that "HUI 700 multiplexes access to the common device between the CPU 702 and the SCP 706" in col. 82, lines 8-9; FIG. 23*) to the non-volatile memory from the processor to the microcontroller (*note "[t]he HUI 700 allows both the CPU 702 and the SCP 706 to share a common memory device 704" in col. 81, lines 40-42*); while the system is held in the first state, fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller (*note "the BIOS, as well as the SCP code, can be stored in a single memory device" in col. 81, lines 42-43, and "[f]ull access is provided to both the CPU 702, as well as the SCP 706, utilizing a time based interleaving method as illustrated in FIG. 24" in col. 81, lines 50-52; FIG. 24*); and after the program instructions or data have been loaded, switch access to the non-volatile memory from the microcontroller to the processor (*note that "HUI 700 multiplexes access to the common device between the CPU 702 and the SCP 706" in col. 82, lines 8-9; FIG. 23*), and release the system from the first state. (*FIG.20; "The HUI solves this problem by providing an interface to enable both the SCP and the CPU to share a common memory device ", col. 36 lines 30-32; "Reads' and writes by the CPU 702 to the HUI700 are under the control of various CPU control signals including an I/O read strobe N\_IORC, an I/O write strobe N\_IOWC, a memory data read strobe N\_MRDC and a memory data write strobe N\_MWTC", col. 36, lines 56-60. "Communication between the SCP 706 and the CPU 702, and between the SCP 706 and the common memory device 704 is under the control of various control signals including a memory read strobe N\_SCPRD, a memory write strobe N\_SCPWR, and*

Art Unit: 3992

*address latch enable N\_SCPALE and a program store enable N\_CPPSE", col. 36-37, lines 66-4; DeRoo also discloses a CONTROL 720 in FIG. 20 for accessing the common memory device 704 from either the CPU register file 712 or the SCP register file 714, col. 37, lines 6-23).*

With regard to claim 2, DeRoo teaches that the change in system state to the first state is performed by software (*note "the HUI 700 allows the SCP 706 to communicate with the CPU 702 and the common memory device 704 by way of an SCP register file 714" in col. 36, lines 42-44).*

With regard to claim 4, DeRoo discloses that the program instructions or data comprise firmware code or data for the microcontroller (*"Write external RAM (program &/or data)" and "Read from external Program/Data memory", col. 22, Table VII, last 2 lines).*

With regard to claim 5, DeRoo teaches that the microcontroller is a peripheral device controller for a computer system (*note "The SCP 20 monitors the keyboard 22" in col. 4, line 22).*

With regard to claim 6, DeRoo discloses a method for sharing a non-volatile memory (*FIG. 20, common memory device 704*) between a processor (*FIG. 20, CPU 702*) and a microcontroller (*FIG. 20, SCP 706 and HUI 700*) in a system (*note the multiprocessor system of FIG. 20*), comprising: in response to a change in system state to a first state (*FIG. 24, when the CPU SYSTEM ACCESS signal is in low level*) wherein the microcontroller is assured safe access to the non-volatile memory, holding the system in the first state (*note address latch enable N\_SCPALE*) and switching access (*note that "HUI 700 multiplexes access to the common device between the CPU 702 and the SCP 706" in col. 82, lines 8-9; FIG. 23*) to the non-volatile

Art Unit: 3992

memory from the processor to the microcontroller (*note "[t]he HUI 700 allows both the CPU 702 and the SCP 706 to share a common memory device 704" in col. 81, lines 40-42*); while the system is held in the first state, fetching program instructions or data from the non-volatile memory and loading the program instructions or data into a memory of the microcontroller (*note "the BIOS, as well as the SCP code, can be stored in a single memory device" in col. 81, lines 42-43, and "[f]ull access is provided to both the CPU 702, as well as the SCP 706, utilizing a time based interleaving method as illustrated in FIG. 24" in col. 81, lines 50-52; FIG. 24*); and after the program instructions or data have been loaded, switching access to the non-volatile memory from the microcontroller to the processor, and releasing the system from the first state (FIG. 20; *"The HUI solves this problem by providing an interface to enable both the SCP and the CPU to share a common memory device ", col. 36 lines 30-32; "Reads' and writes by the CPU 702 to the HUI 700 are under the control of various CPU control signals including an I/O read strobe N\_IORC, an I/O write strobe N\_IOWC, a memory data read strobe N\_MRDC and a memory data write strobe N\_MWTC", col. 36, lines 56-60. "Communication between the SCP 706 and the CPU 702, and between the SCP 706 and the common memory device 704 is under the control of various control signals including a memory read strobe N\_SCPRD, a memory write strobe N\_SCPWR, and address latch enable N\_SCPALE and a program store enable N\_CPPSE", col. 36-37, lines 66-4. Moreover, DeRoo also discloses a CONTROL 720 in FIG. 20 for accessing the common memory device 704 from either the CPU register file 712 or the SCP register file 714, col. 37, lines 6-23 ).*

With regard to claim 7, DeRoo discloses that the change in system state to the first state is performed by software (note that *"the HUI 700 allows the SCP 706 to communicate with the*

*CPU 702 and the common memory device 704 by way of an SCP register file 714" in col. 36, lines 42-44).*

With regard to claim 9, DeRoo discloses the program instructions or data comprise firmware code or data for the microcontroller ("*Write external RAM (program &/or data)*" and "*Read from external Program/Data memory*", col. 22, Table VII, last 2 lines).

With regard to claim 10, DeRoo discloses the microcontroller is a peripheral device controller for a computer system (note "*The SCP 20 monitors the keyboard 22*" in col. 4, line 22).

## **Issue 2**

*A SNQ as to claims 3, 8 and 11-23 is raised by DeRoo.*

9. Claims 3, 8 and 11-23 are rejected under 35 U.S.C. § 103(a) as being unpatentable over DeRoo.

With regard to claims 3 and 8, DeRoo does not explicitly teach that the first state comprises one or more of a power on reset (POR) state, a system reset state or a sleep state. However, powering on, resetting and sleep mode were well known initial states for memory system operation. Thus, it would have been obvious to one having ordinary skill in the art at the time the invention was made to utilize one or more of a power on reset (POR) state, a system reset state or a sleep state as the initial state, since (a) a power on reset state or a system reset state would have allowed the most current system status, parameters and drivers to be loaded for



Art Unit: 3992

more efficient operation and, alternatively, (b) using a sleep state as an initial state would have the advantage of saving power while still retaining the memory state and without requiring rebooting of the operating system.

With regard to claim 11, DeRoo discloses a system, comprising: a non-volatile memory (*FIG 20, common memory device 704*); a processor (*FIG20, CPU 702*); and a microcontroller (*FIG. 20, SCP 706 and HUI 700*) coupled to the non-volatile memory and to the processor.

DeRoo does not explicitly teach that the microcontroller is configured to: in response to a power on reset, hold a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory; while the system reset signal is held in the reset state, fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller; and after the program instructions or data have been loaded, permit the processor in the system access to the non-volatile memory, and release the system reset signal from the reset state.

Even though DeRoo does not explicitly teach holding the processor in a *reset* state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so, DeRoo does teach (1) holding a processor in a *wait* state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so (*note col. 81, lines 52-64*) and (2) holding the microcontroller in a reset state to prohibit the microcontroller from accessing the non-volatile memory, while allowing the processor to do so (*note, e.g., col. 81, line 64, to column 82, line 6*).

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to hold the processor in the system of DeRoo in a reset state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller exclusive access to the non-volatile memory until the reset signal is released, since (1) DeRoo specifically taught the use of a reset state to prevent an element from accessing the non-volatile memory; namely, holding the microcontroller in a reset state to prohibit the microcontroller from accessing the non-volatile memory, while allowing the processor to do so, (2) DeRoo specifically taught the use of a state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller exclusive access to the non-volatile memory; namely, holding a processor in a wait state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so and (3) it may have been advantageous to take make use of any idle time during the wait state of the processor by performing a reset operation in the processor to run new updates, etc.

With regard to claim 12, DeRoo discloses the program instructions or data comprise firmware code or data for the microcontroller ("*Write external RAM (program &/or data)*" and "*Read from external Program/Data memory*", col. 22, Table VII, last 2 lines).

With regard to claim 13, DeRoo discloses the microcontroller is a peripheral device controller for a computer system (note "*The SCP 20 monitors the keyboard 22*" in col. 4, line 22).

With regard to claim 14, DeRoo discloses the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to

Art Unit: 3992

fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus (*FIG. 20, FD[0:7]*).

With regard to claim 15, DeRoo discloses the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable to switchably intercept or pass communications between the processor and the non-volatile memory (*FIGs. 20 and 23, address mux 716, data mux & latch 718*).

With regard to claim 16, DeRoo discloses the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor and the non-volatile memory (*FIGs. 20 and 23, address mux 716, data mux & latch 718; note "the SCP 706 can gain exclusive access to the common memory device 704 by asserting a CPU EXCLUDED signal (i.e., setting bit i of the flash interface control register FLASH\_CTRL). During such a condition, the data multiplexer 718 blocks the system data bus SD[0:7] from the path of the memory data bus FD[0:7]" in col. 82, lines 32-39; the address mux 716 and the data mux and latch 718 are controlled by the outputs of control logic 720, and the values of the registers in the control logic 720 can be set by either SCP or CPU*).

With regard to claim 17, DeRoo discloses a method for sharing a non-volatile memory (*FIG 20, common memory device 704*) between a processor (*FIG20, CPU 702*) and a microcontroller (*FIG. 20, SCP 706 and HUI 700*) in a system.

DeRoo does not explicitly teach in response to a power on reset, holding a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory; while the system reset signal is held in the reset state, fetching program instructions or data from the non-volatile memory and loading the program instructions or data into a memory of the microcontroller; and after the program instructions or data have been loaded, permitting the processor in the system access to the non-volatile memory, and releasing the system reset signal from the reset state.

Even though DeRoo does not explicitly teach holding the processor in a *reset* state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so, DeRoo does teach (1) holding a processor in a *wait* state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so (*note col. 81, lines 52-64*) and (2) holding the microcontroller in a reset state to prohibit the microcontroller from accessing the non-volatile memory, while allowing the processor to do so (*note, e.g., col. 81, line 64, to column 82, line 6*).

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to hold the processor in the system of DeRoo in a reset state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller exclusive access to the non-volatile memory until the reset signal is released, since (1) DeRoo specifically taught the use of a reset state to prevent an element from accessing the non-volatile memory; namely, holding the microcontroller in a reset state to prohibit the microcontroller from accessing the non-volatile memory, while allowing the processor to do so, (2) DeRoo specifically taught the use of a state to prohibit the processor from accessing the non-volatile

Art Unit: 3992

memory, while allowing the microcontroller exclusive access to the non-volatile memory; namely, holding a processor in a wait state to prohibit the processor from accessing the non-volatile memory, while allowing the microcontroller to do so and (3) it may have been advantageous to take make use of any idle time during the wait state of the processor by performing a reset operation in the processor to run new updates, etc.

With regard to claim 18, DeRoo discloses the program instructions or data comprise firmware code or data for the microcontroller (*"Write external RAM (program &/or data)" and "Read from external Program/Data memory", col. 22, Table VII, last 2 lines*).

With regard to claim 19, DeRoo discloses the microcontroller is a peripheral device controller for a computer system (*note "The SCP 20 monitors the keyboard 22" in col. 4, line 22*).

With regard to claim 20, DeRoo discloses the microcontroller is coupled to the processor via a system reset bus, and wherein the microcontroller is operable to assert the system reset signal over the system reset bus (*FIG. 20, RESET line for SCP*).

With regard to claim 21, DeRoo discloses the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus (*FIGs. 20 and 23, FD[0:7]*).

With regard to claim 22, DeRoo discloses the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable switchably intercept or pass

Art Unit: 3992

communications between the processor and the non-volatile memory (*FIGs. 20 and 23, address mux 716, data mux & latch 718*).

With regard to claim 23, DeRoo teaches that the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor and the non-volatile memory (*FIGs. 20 and 23, address mux 716, data mux & latch 718; note "the SCP 706 can gain exclusive access to the common memory device 704 by asserting a CPU EXCLUDED signal (i.e., setting bit i of the flash interface control register FLASH\_CTRL). During such a condition, the data multiplexer 718 blocks the system data bus SD[0:7] from the path of the memory data bus FD[0:7]" in col. 82, lines 32-39; the address mux 716 and the data mux and latch 718 are controlled by the outputs of control logic 720, and the values of the registers in the control logic 720 can be set by either SCP or CPU*).

***Examiner's Statement of Reasons for Patentability/Confirmation***

10. None of the claims subject to reexamination have been confirmed at this time.

## CONCLUSION

### *Amendment in Reexamination Proceedings*

11. Patent Owner is notified that any proposed amendment to the specification and/or claims in this reexamination proceeding must comply with 37 CFR 1.530(d)-(j), must be formally presented pursuant to 37 CFR 1.52(a) and (b), and must contain any fees required by 37 CFR 1.20(c).

In order to ensure full consideration of any amendments, affidavits or declarations, or other documents as evidence of patentability, such documents must be submitted in response to this Office action. Submissions will be governed by the requirements of 37 CFR 1.116, after final rejection and 37 CFR 41.33 after appeal, which will be strictly enforced. See MPEP § 2250(IV) for examples to assist in the preparation of proper proposed amendments in reexamination proceedings.

### *Service of Papers*

12. After filing of a request for ex parte reexamination by a third party requester, any document filed by either the patent owner or the third party requester must be served on the other party (or parties where two or more third party requester proceedings are merged) in the reexamination proceeding in the manner provided in 37 CFR 1.248. The document must reflect service or the document may be refused consideration by the Office. See 37 CFR 1.550(f).

***Extensions of Time***

13. Extensions of time under 37 CFR 1.136(a) will not be permitted in these proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Additionally, 35 U.S.C. 305 requires that *ex parte* reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.550(a)). Extensions of time in *ex parte* reexamination proceedings are provided for in 37 CFR 1.550(c).

***Litigation Reminder***

14. The patent owner is reminded of the continuing responsibility under 37 CFR 1.565(a) to apprise the Office of any litigation activity, or other prior or concurrent proceeding, involving the patent throughout the course of this reexamination proceeding. The third party requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §§ 2207, 2282 and 2286.



Art Unit: 3992

All correspondence relating to this *ex parte* reexamination proceeding should be directed:

By Mail to: Mail Stop *Ex Parte* Reexam  
Central Reexamination Unit  
Commissioner for Patents  
United States Patent & Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450

By FAX to: (571) 273-9900  
Central Reexamination Unit

By hand to: Customer Service Window  
Randolph Building  
401 Dulany Street  
Alexandria, VA 22314

Registered users of EFS-Web may alternatively submit such correspondence via the electronic filing system EFS-Web, at <https://efs.uspto.gov/efile/myportal/efs-registered>. EFS-Web offers the benefit of quick submission to the particular area of the Office that needs to act on the correspondence. Also, EFS-Web submissions are "soft scanned" (i.e., electronically uploaded) directly into the official file for the reexamination proceeding, which offers parties the opportunity to review the content of their submissions after the "soft scanning" process is complete.

Any inquiry concerning this communication should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

Signed:

/B. James Peikari/

B. James Peikari  
Primary Examiner  
Art Unit 3992

Conferees:  
/EBK/



MARK J. REINHART  
CRU SPE-AU 3992



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| APPLICATION NO.  | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|--|-------------|----------------------|---------------------|------------------|
| 90/011,754   | 06/21/2011  | 7,930,576 B2         | JCLA38862-RE        | 4071             |
| 35690  | 7590        | 09/08/2011           | EXAMINER            |                  |
| MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.<br>P.O. BOX 398<br>AUSTIN, TX 78767-0398 |             |                      | ART UNIT            | PAPER NUMBER     |

DATE MAILED: 09/08/2011

Please find below and/or attached an Office communication concerning this application or proceeding.



UNITED STATES PATENT AND TRADEMARK OFFICE

---

Commissioner for Patents  
United States Patents and Trademark Office  
P.O.Box 1450  
Alexandria, VA 22313-1450  
www.uspto.gov

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS

JIA WEI HUANG (J.C. PATENTS)

4, VENTURE

SUITE 250

IRVINE, CA 92618

Date:

**MAILED**

SEP 08 2011

**CENTRAL REEXAMINATION UNIT**

**EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM**

REEXAMINATION CONTROL NO. : 90011754

PATENT NO. : 7930576

ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified ex parte reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the ex parte reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

---

|  |                                   |  |  |
|--|-----------------------------------|--|--|
| <b>Order Granting / Denying Request For<br/>Ex Parte Reexamination</b> | <b>Control No.</b><br>90/011,754  | <b>Patent Under Reexamination</b><br>7,930,576 B2 ET |  |
|  | <b>Examiner</b><br>BEHZAD PEIKARI | <b>Art Unit</b><br>3992                              |  |

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

The request for *ex parte* reexamination filed 21 June 2011 has been considered and a determination has been made. An identification of the claims, the references relied upon, and the rationale supporting the determination are attached.

Attachments: a) ☐ PTO-892, b) ☐ PTO/SB/08, c) ☒ Other: PTO-1449

1. ☒ The request for *ex parte* reexamination is GRANTED.

**RESPONSE TIMES ARE SET AS FOLLOWS:**

For Patent Owner's Statement (Optional): TWO MONTHS from the mailing date of this communication (37 CFR 1.530 (b)). **EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.550(c).**

For Requester's Reply (optional): TWO MONTHS from the **date of service** of any timely filed Patent Owner's Statement (37 CFR 1.535). **NO EXTENSION OF THIS TIME PERIOD IS PERMITTED.** If Patent Owner does not file a timely statement under 37 CFR 1.530(b), then no reply by requester is permitted.

2. ☐ The request for *ex parte* reexamination is DENIED.

This decision is not appealable (35 U.S.C. 303(c)). Requester may seek review by petition to the Commissioner under 37 CFR 1.181 within ONE MONTH from the mailing date of this communication (37 CFR 1.515(c)). **EXTENSION OF TIME TO FILE SUCH A PETITION UNDER 37 CFR 1.181 ARE AVAILABLE ONLY BY PETITION TO SUSPEND OR WAIVE THE REGULATIONS UNDER 37 CFR 1.183.**

In due course, a refund under 37 CFR 1.26 ( c ) will be made to requester:

- a) ☐ by Treasury check or,  
b) ☐ by credit to Deposit Account No. \_\_\_\_\_, or  
c) ☐ by credit to a credit card account, unless otherwise notified (35 U.S.C. 303(c)).

cc:Requester ( if third party requester )

U.S. Patent and Trademark Office  
PTOL-471 (Rev. 08-06)

Office Action in *Ex Parte* Reexamination

Part of Paper No. 20110823

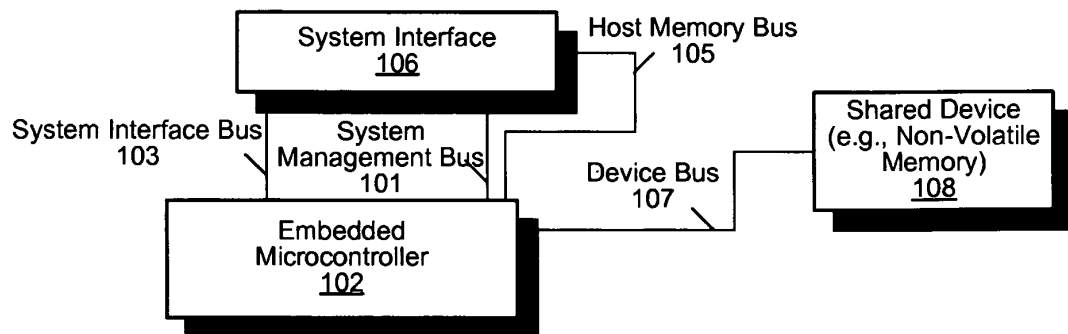
## RESPONSE TO REQUEST FOR *EX PARTE* REEXAMINATION

1. The Request filed by a third part requestor on June 21, 2011 alleges that there is a substantial new question of patentability (SNQ) affecting claims 1-23 of U.S. Patent Number 7,930,576 (the '576 patent) based on the following prior art reference:

- U.S. Patent No. 6,161,162 to DeRoo et al. ("DeRoo").

### *Brief Overview of the Patent*

2. The '576 patent is directed to a system and method for sharing a non-volatile memory between a host processor and a microcontroller.



In response to system state change to a first state wherein the microcontroller is assured safe access to the memory, the microcontroller holds the system in the first state and obtains access to the memory, accesses the memory and loads program instructions and/or data into a memory of the microcontroller. Figure 1b of the patent is shown above.

After the access, the microcontroller changes or allows change of the system state from the microcontroller to the processor, and releases the system from the first state.

### ***Prosecution History***

3. The '576 patent matured from U.S. Patent Application No. 11/958,601, whose relevant prosecution history may be summarized as follows:

- The application was filed on December 18, 2007 with claims 1-25.
- In a non-final Office Action mailed September 3, 2010, claims 1-23 were rejected under 35 U.S.C. §112, second paragraph, claims 24 and 25 were rejected under 35 U.S.C. §102(b) as being anticipated by Le et al., U.S. Patent No. 6,154,838, and claims 1-23 were rejected under 35 U.S.C. §103 as being obvious over Le et al., U.S. Patent No. 6,154,838, in view of Pang et al., U.S. Patent No. 6,925,522.
- In an amendment filed December 3, 2010, claims 1, 4, 6, 9, 11, 12, 14, 16, 17, 18, and 21 were amended and claims 24 and 25 were cancelled.
- On December 17, 2010 a Notice of Allowance was mailed, in which the examiner allowed claims 1-23.
- On April 19, 2011 the application issued as U.S. Patent No. 7,930,576.

### ***SNQs Raised in the Request***

4. The Requester identified the following SNQs (see Request, page 6):

- **Issue 1:** A SNQ as to claims 1-2, 4-7 and 9-10 is raised by DeRoo.
- **Issue 2:** A SNQ as to claims 3, 8 and 11-23 is raised by DeRoo.

*Discussion of SNQs*

5. A prior art patent or printed publication raises a substantial new question of patentability where there is:

(A) a substantial likelihood that a reasonable Examiner would consider the prior art patent or printed publication important in deciding whether or not the claim is patentable, MPEP §2242 (I) and,

(B) the same question of patentability as to the claim has not been decided in a previous or pending proceeding or in a final holding of invalidity by a federal court. See MPEP §2242 (III).

For any reexamination ordered on or after November 2, 2002, reliance on previously cited/considered art, i.e., "old art," does not necessarily preclude the existence of a substantial new question of patentability that is based exclusively on that old art. Rather, determinations on whether a substantial new question of patentability exists in such an instance shall be based upon a fact-specific inquiry done on a case-by-case basis. See MPEP 2242.

**Issue 1**

*A SNQ as to claims 1-2, 4-7 and 9-10 is raised by DeRoo.*

6. DeRoo discloses a multiprocessor system that enables shared access to a memory by multiple CPUs (*note Figure 20 of DeRoo*).

Multiplexed address and data paths are provided from three CPUs to the single storage device. The paths are controlled by an arbitration circuit which allows one CPU to always have the highest priority. The primary CPU may or may not be the highest priority CPU in the arbitration scheme. The arbitration circuit is combined with a controller which interfaces to the memory device.

The controller is responsive to these processors to multiplex their information signals for selectively conveying information present at their address and data ports to ensure that the memory device is addressable by the processors, and responsive to the controller to share addressing of the common memory device.

Thus DeRoo discloses a system for shared access to a memory by a primary CPU (e.g., host CPU) and another controlling device, which it appears the examiner considered to be allowable features of claims 1-2, 4-7 and 9-10.

7. There is a substantial likelihood that a reasonable examiner would consider the teachings of DeRoo important in deciding the patentability of claims 1-2, 4-7 and 9-10 of the '576 patent. DeRoo is not of record in the file of the '106 patent and is not cumulative to the art of record in the original file. The teachings of DeRoo were not subject to a final holding of invalidity by a federal court.

Accordingly, it is **agreed** that the DeRoo an SNQ as to claims 1-2, 4-7 and 9-10.



**Issue 2**

*A SNQ as to claims 3, 8 and 11-23 is raised by DeRoo.*

8. The system of DeRoo has been described above.

Thus DeRoo discloses a system for shared access to a memory by a primary CPU (e.g., host CPU) and another controlling device. The request presents arguments regarding the first state comprising one or more of: a power on reset (POR) state; a system reset state; or a sleep state, which it appears the examiner considered to be allowable features of claims 3, 8 and 11-23.

9. There is a substantial likelihood that a reasonable examiner would consider the combination of DeRoo important in deciding the patentability of the claims of the '576 patent.

Accordingly, it is **agreed** that DeRoo raises a substantial new question of patentability as to claims 3, 8 and 11-23.

***Conclusion***

10. There is at least one substantially new question of patentability for each of claims 1-23.

Thus, claims 1-23 are subject to reexamination.

***Waiver of Right to File Patent Owner Statement***

11. In a reexamination proceeding, Patent Owner may waive the right under 37 C.F.R. 1.530 to file a Patent Owner Statement. The document needs to contain a statement that Patent Owner waives the right under 37 C.F.R. 1.530 to file a Patent Owner Statement and proof of service in the manner provided by 37 C.F.R. 1.248, if the request for reexamination was made by a third party requester, see 37 C.F.R. 1.550(f).

***Service of Papers***

12. After filing of a request for ex parte reexamination by a third party requester, any document filed by either the patent owner or the third party requester must be served on the other party (or parties where two or more third party requester proceedings are merged) in the reexamination proceeding in the manner provided in 37 CFR 1.248. The document must reflect service or the document may be refused consideration by the Office. See 37 CFR 1.550(f).

***Extensions of Time***

13. Extensions of time under 37 CFR 1.136(a) will not be permitted in these proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Additionally, 35 U.S.C. 305 requires that *ex parte* reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.550(a)). Extensions of time in *ex parte* reexamination proceedings are provided for in 37 CFR 1.550(c).

***Litigation Reminder***

14. The patent owner is reminded of the continuing responsibility under 37 CFR 1.565(a) to apprise the Office of any litigation activity, or other prior or concurrent proceeding, involving Patent No. 6,119,181 throughout the course of this reexamination proceeding. The third party requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §§ 2207, 2282 and 2286.

Art Unit: 3992

All correspondence relating to this *ex parte* reexamination proceeding should be directed:

By Mail to: Mail Stop *Ex Parte* Reexam  
Central Reexamination Unit  
Commissioner for Patents  
United States Patent & Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450

By FAX to: (571) 273-9900  
Central Reexamination Unit

By hand to: Customer Service Window  
Randolph Building  
401 Dulany Street  
Alexandria, VA 22314

Registered users of EFS-Web may alternatively submit such correspondence via the electronic filing system EFS-Web, at <https://sportal.uspto.gov/authenticate/authenticateuserlocalepf.html>. EFS-Web offers the benefit of quick submission to the particular area of the Office that needs to act on the correspondence. Also, EFS-Web submissions are "soft scanned" (i.e., electronically uploaded) directly into the official file for the reexamination proceeding, which offers parties the opportunity to review the content of their submissions after the "soft scanning" process is complete.

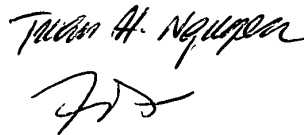
Any inquiry concerning this communication should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

Signed:

Conferees:

/B. James Peikari/

B. James Peikari  
Primary Examiner  
Art Unit 3992



Handwritten signature of T. H. Nguyen, with a second signature below it.

Receipt date: 06/21/2011

90011754 - GAU: 3992

|   |                                 |                    |
|---|---------------------------------|--------------------|
| FORM PTO-1449 U.S. DEPARTMENT OF COMMERCE<br>PATENT AND TRADEMARK OFFICE<br><br><b>INFORMATION DISCLOSURE STATEMENT BY<br/>         APPLICANT</b> | ATTY. DOCKET NO.: JCLA38862-RE  | APPLICATION No.: / |
|   | APPLICANT: IAN F. HARRIS et al. |                    |
|   | FILING DATE: HERewith           | GROUP:             |

**U.S. PATENT DOCUMENTS**

| EXAMINER<br>INITIAL |   | DOCUMENT NUMBER | DATE       | NAME                  | CLASS | SUBCLASS | FILING DATE<br>(IF APPROPRIATE) |
|---------------------|---|-----------------|------------|-----------------------|-------|----------|---------------------------------|
| /BJP/               | 1 | 6,161,162       | 12/12/2000 | David T. DeRoo et al. | 710   | 244      |                                 |
|                     |   |                 |            |                       |       |          |                                 |
|                     |   |                 |            |                       |       |          |                                 |
|                     |   |                 |            |                       |       |          |                                 |
|                     |   |                 |            |                       |       |          |                                 |
|                     |   |                 |            |                       |       |          |                                 |

**FOREIGN PATENT DOCUMENTS**


| EXAMINER<br>INITIAL |  | DOCUMENT NUMBER | PUBLICATION<br>DATE | COUNTRY | CLASS | SUBCLASS | ENGLISH<br>TRANSLATION |
|---------------------|--|-----------------|---------------------|---------|-------|----------|------------------------|
|                     |  |                 |                     |         |       |          |                        |
|                     |  |                 |                     |         |       |          |                        |
|                     |  |                 |                     |         |       |          |                        |

**NON PATENT LITERATURE DOCUMENTS**

| EXAMINER<br>INITIAL |  | Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or county where published. | ENGLISH<br>TRANSLATION |
|---------------------|--|--|------------------------|
|                     |  |  |                        |
|                     |  |  |                        |
|                     |  |  |                        |

|                                |                               |
|--------------------------------|-------------------------------|
| EXAMINER<br>/B. James Peikari/ | DATE CONSIDERED<br>08/25/2011 |
|--------------------------------|-------------------------------|


**EXAMINER:** INITIAL IF CITATION CONSIDERED, WHETHER OR NOT CITATION IS IN CONFORMANCE WITH MPEP 609; DRAW LINE THROUGH CITATION IF NOT IN CONFORMANCE AND NOT CONSIDERED. INCLUDE COPY OF THIS FORM WITH NEXT COMMUNICATION TO APPLICANT.

|   |                         |   |
|---|-------------------------|---|
| <b>Reexamination</b><br> | Application/Control No. | Applicant(s)/Patent Under Reexamination |
|   | 90/011,754              | 7,930,576 B2 ET AL.                     |
|   | Certificate Date        | Certificate Number                      |

|   |                         |                                       |   |
|---|-------------------------|---------------------------------------|---|
| Requester   | Correspondence Address: | <input type="checkbox"/> Patent Owner | <input checked="" type="checkbox"/> Third Party |
| <p>MEYERTONS, HOOD, KIVLIN, KOWERT &amp; GOETZEL, P.C.<br/> P.O. BOX 398<br/> AUSTIN, TX 78767-0398</p> |                         |                                       |   |

|   |                            |                   |
|---|----------------------------|-------------------|
| LITIGATION REVIEW <input checked="" type="checkbox"/> | BJP<br>(examiner initials) | 8/24/11<br>(date) |
| Case Name   |                            | Director Initials |
| None  |                            | <i>For Entry</i>  |
|   |                            |                   |
|   |                            |                   |
|   |                            |                   |
|   |                            |                   |

| COPENDING OFFICE PROCEEDINGS |        |
|------------------------------|--------|
| TYPE OF PROCEEDING           | NUMBER |
| 1. None                      |        |
| 2.                           |        |
| 3.                           |        |
| 4.                           |        |

|  |                                |  |  |
|--|--------------------------------|--|--|
| <b>Search Notes</b><br> | <b>Application/Control No.</b> | <b>Applicant(s)/Patent under Reexamination</b> |  |
|  | 90/011,754                     | 7,930,576 B2 ET AL.                            |  |
|  | <b>Examiner</b>                | <b>Art Unit</b>                                |  |
|  | BEHZAD PEIKARI                 | 3992   |  |

| SEARCHED |          |      |          |
|----------|----------|------|----------|
| Class    | Subclass | Date | Examiner |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |
|          |          |      |          |

| INTERFERENCE SEARCHED |          |      |          |
|-----------------------|----------|------|----------|
| Class                 | Subclass | Date | Examiner |
|                       |          |      |          |
|                       |          |      |          |
|                       |          |      |          |
|                       |          |      |          |
|                       |          |      |          |

| SEARCH NOTES<br>(INCLUDING SEARCH STRATEGY)                     |           |      |
|---|-----------|------|
|   | DATE      | EXMR |
| Review of the prosecution history of U.S. Patent No. 7,930,576. | 8/24/2011 | BJP  |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |
|   |           |      |



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| APPLICATION NO.  | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|--|-------------|----------------------|---------------------|------------------|
| 90/011,754   | 06/21/2011  | 7,930,576 B2         | JCLA38862-RE        | 4071             |
| 35690  | 7590        | 08/29/2011           | EXAMINER            |                  |
| MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.<br>P.O. BOX 398<br>AUSTIN, TX 78767-0398 |             |                      | ART UNIT            | PAPER NUMBER     |

DATE MAILED: 08/29/2011

Please find below and/or attached an Office communication concerning this application or proceeding.





UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents  
United States Patents and Trademark Office  
P.O.Box 1450  
Alexandria, VA 22313-1450  
www.uspto.gov

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS

JIAWEI HUANG (J.C. PATENTS)

4, VENTURE

SUITE 250

IRVINE, CA 92618

Date:

**MAILED**

**AUG 29 2011**

**CENTRAL REEXAMINATION UNIT**

**EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM**

REEXAMINATION CONTROL NO. : 90011754

PATENT NO. : 7930576

ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified ex parte reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the ex parte reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

|  |             |  |
|--|-------------|--|
| <b>Ex Parte Reexamination Interview<br/>Summary – Pilot Program for Waiver of<br/>Patent Owner's Statement</b> | Control No. | Patent For Which Reexamination<br>is Requested |
|  | 90/011,754  | 7,930,576                                      |
|  | Examiner    | Art Unit                                       |
|  |             | 3992   |

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --

All participants (USPTO official and patent owner):

**MAILED**

(1) Karen Ward, CRU

(3)

AUG 29 2011

(2) Eric Meyertons, 34876

(4)

**CENTRAL REEXAMINATION UNIT**

Date of Telephonic Interview: August 25, 2011.

The USPTO official requested waiver of the patent owner's statement pursuant to the pilot program for waiver of patent owner's statement in *ex parte* reexamination proceedings.\*

☐ The patent owner **agreed** to waive its right to file a patent owner's statement under 35 U.S.C. 304 in the event reexamination is ordered for the above-identified patent.

☐ The patent owner **did not agree** to waive its right to file a patent owner's statement under 35 U.S.C. 304 at this time.

The patent owner is not required to file a written statement of this telephone communication under 37 CFR 1.560(b) or otherwise. However, any disagreement as to this interview summary must be brought to the immediate attention of the USPTO, and no later than one month from the mailing date of this interview summary. Extensions of time are governed by 37 CFR 1.550(c).

\*For more information regarding this pilot program, see *Pilot Program for Waiver of Patent Owner's Statement in Ex Parte Reexamination Proceedings*, 75 Fed. Reg. 47269 (August 5, 2010), available on the USPTO Web site at <http://www.uspto.gov/patents/law/notices/2010.jsp>.

☒ USPTO personnel were unable to reach the patent owner.

The patent owner may contact the USPTO personnel at the telephone number provided below if the patent owner decides to waive the right to file a patent owner's statement under 35 U.S.C. 304.

/Karen Ward/

571-272-7932

Signature and telephone number of the USPTO official who contacted or attempted to contact the patent owner.

cc: Requester (if third party requester)

# Litigation Search Report CRU 3999

Reexam Control No. 90/011,754

**To: Behzad Peikari**

**Art Unit: 3992**

**Date: 08/19/11**

**Case Serial Number: 90/011,754**

**From: Karen L. Ward**

**Location: CRU 3999**

**MDW 7C76**

**Phone: (571) 272-7932**

**Karen.Ward@uspto.gov**

## Search Notes

Litigation was not found involving U.S. Patent No. 7,930,576.

- 1) I performed a KeyCite Search in Westlaw, which retrieves all history on the patent including any litigation.
- 2) I performed a search on the patent in Lexis CourtLink for any open dockets or closed cases.
- 3) I performed a search in Lexis in the Federal Courts and Administrative Materials databases for any cases found.
- 4) I performed a search in Lexis in the IP Journal and Periodicals database for any articles on the patent.
- 5) I performed a search in Lexis in the news databases for any articles about the patent or any articles about litigation on this patent.



Date of Printing: Aug 19, 2011

# KEYCITE

**C** US PAT 7930576 SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM, Assignee: Standard Microsystems Corporation (Apr 19, 2011)

## History

### Direct History

=> 1 SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM, US PAT 7930576, 2011 WL 1470299 (U.S. PTO Utility Apr 19, 2011) (NO. 11/958601)

## Patent Family

2 SYSTEM FOR SHARING NON-VOLATILE MEMORY IN COMPUTER SYSTEM, HAS MICROCONTROLLER THAT CONTROLS ACCESS OF SELF DEVICE AND MAIN PROCESSOR TO NON-VOLATILE MEMORY IN RESPONSE TO CHANGE IN SYSTEM STATE, Derwent World Patents Legal 2008-M01067

## Assignments

3 Action: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS). Number of Pages: 006, (DATE RECORDED: Dec 18, 2007)

## Patent Status Files

.. Request for Re-Examination, (OG DATE: Aug 09, 2011)

## Prior Art (Coverage Begins 1976)

- C** 5 CIRCUITRY AND METHOD FOR SHARING INTERNAL MICROCONTROLLER MEMORY WITH AN EXTERNAL PROCESSOR, US PAT 5428760 Assignee: Intel Corporation, (U.S. PTO Utility 1995)
- C** 6 DEVICE AND METHOD CAPABLE OF CHANGING CODES OF MICRO-CONTROLLER, US PAT 6925522 Assignee: Lite-On It Corporation, (U.S. PTO Utility 2005)
- C** 7 DIRECTLY PROGRAMMABLE DISTRIBUTION ELEMENT, US PAT 5925097 Assignee: Network Machines, Inc., (U.S. PTO Utility 1999)
- C** 8 DIRECTLY PROGRAMMABLE DISTRIBUTION ELEMENT, US PAT 5634004 Assignee: Network Programs, Inc., (U.S. PTO Utility 1997)
- C** 9 FLASH ROM SHARING BETWEEN PROCESSOR AND MICROCONTROLLER DURING BOOTING AND HANDLING WARM-BOOTING EVENTS, US PAT 6154838 (U.S. PTO Utility 2000)

© 2011 Thomson Reuters. All rights reserved.

- C** 10 FLASH ROM SHARING BETWEEN PROCESSOR AND MICROCONTROLLER DURING BOOTING AND HANDLING WARM-BOOTING EVENTS, US PAT 5819087Assignee: Compaq Computer Corporation, (U.S. PTO Utility 1998)
- C** 11 METHOD AND APPARATUS FOR PROGRAMMING A MICROPROCESSOR USING AN ADDRESS DECODE CIRCUIT, US PAT 5949997Assignee: NCR Corporation, (U.S. PTO Utility 1999)
- C** 12 PROGRAMMABLE TASK-BASED CO-PROCESSOR, US PAT 7386326Assignee: Texas Instruments Incorporated, (U.S. PTO Utility 2008)

[My Briefcase](#) | [Order Runner Documents](#) | [Available Courts](#) | [Learning Center](#)

**Single Search - with Terms and Connectors**

Enter keywords - Search multiple dockets & documents

**Search**

[View Demo](#)  
[Search Tips](#)

[My CourtLink](#) [Search](#) [Dockets & Documents](#) [Track](#) [Alert](#) [Strategic Profiles](#) [My Account](#)



[Search](#) > [Patent Search](#) > Searching

Patent Search 7930576 8/19/2011

No cases found.

[Return to Search](#)

(Charges for search still apply)





**LexisNexis®**

[About LexisNexis](#) | [Terms & Conditions](#) | [Pricing](#) | [Privacy](#) | [Customer Support](#) - 1-888-311-1966  
Copyright © 2011 LexisNexis®. All rights reserved.

[Switch Client](#) | [Preferences](#) | [Help](#) | [Sign Out](#)

| <b>My Lexis™</b> | <b>Search</b> | <b>Get a Document</b> | <b>Shepard's®</b> | <b>More</b> | <b>History</b> | <b>Alerts</b> |
|------------------|---------------|-----------------------|-------------------|-------------|----------------|---------------|
|------------------|---------------|-----------------------|-------------------|-------------|----------------|---------------|

**FOCUS™** Terms  Search Within  ☐ Using Semantic  
 Concepts What's this?  [Advanced...](#)

[View Tutorial](#)Source: **Command Searching > Utility, Design and Plant Patents** Terms: **7930576** (Suggest Terms for My Search) Select for FOCUS™ or Delivery

- ☐ 1. 7930576, April 19, 2011, Sharing non-sharable devices between an embedded controller and a processor in a computer system, Harris, Ian F., Kings Park, NEW YORK, United States of America(US), United States of America(US); Dutton, Drew J., Austin, TEXAS, United States of America(US), United States of America(US); 958601, December 18, 2007, ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS)., STANDARD MICROSYSTEMS CORPORATION, 80 ARKAY DRIVE, HAUPPAUGE, NEW YORK, UNITED STATES OF AMERICA(US), 11788, reel-frame:020262/0609, Standard Microsystems Corporation, Hauppauge, NEW YORK, United States of America(US), United States company or corporation

**CORE TERMS:** microcontroller, memory, non-volatile, reset, processor, interface, bus, switch, software, embedded, host, controller, sharing, safe, computer system, hardware, fetch, flash, functionality, configured, exemplary, assured, control signals, handshake, firmware, token, load, preempt, couple, sleep

**7930576**

- ☐ 2. 5687175, November 11, 1997, Adaptive time-division multiplexing communications protocol method and system, Rochester, Virgil Maurice, Las Vegas, United States of America(US); Hunt, Ariel Mark, Vista, United States of America(US); Jonson, Charles Stephen, Las Vegas, United States of America(US); Weaver, John D., Las Vegas, United States of America(US); 341015, April 10, 1995, ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS)., UTICS CORPORATION 9457 LAS VEGAS BLVD SOUTH SUITE GLAS VEGAS, NEVADA, 89123, reel-frame:007424/0951, Utics Corporation, Las Vegas, NEVADA, United States of America(US)

**CORE TERMS:** sensor, packet, mobile unit, transmission, poll, remote, slot, collection, transponder, protocol, send, mobile, frequency, transmitting, platform, transceiver, broadcast, route, requesting, allocated, equipped, data transmission, transmitted, collecting, identifier, monitoring, stationary, message, central station, multiplexing

August 13, 1992 - **07930576**, United States of America (US)Source: **Command Searching > Utility, Design and Plant Patents** Terms: **7930576** (Suggest Terms for My Search)View: [Cite](#)

Date/Time: Friday, August 19, 2011 - 11:01 AM EDT

In

[About LexisNexis](#) | [Privacy Policy](#) | [Terms & Conditions](#) | [Contact Us](#)  
 Copyright © 2011 LexisNexis, a division of Reed Elsevier Inc. All rights reserved.

[Switch Client](#) | [Preferences](#) | [Help](#) | [Sign Out](#)

|                  |               |                            |                       |                      |                  |               |
|------------------|---------------|----------------------------|-----------------------|----------------------|------------------|---------------|
| <b>My Lexis™</b> | <b>Search</b> | <b>Get a Document</b>      | <b>Shepard's®</b>     | <b>More</b>          | <b>History</b>   | <b>Alerts</b> |
| <b>All</b>       | <b>Legal</b>  | <b>News &amp; Business</b> | <b>Public Records</b> | <b>Find A Source</b> | Add/Edit Subtabs |               |

 Command Searching > Patent Cases from Federal Courts and Administrative Materials [i](#)

|   |   |  |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
|---|---|--|-------------------|-----|-------------------|----|----|-------|-----------------|-----|----------------|-----|------------------|-------|---------------------|---------|---------|---|
| <b>Search</b>   |   | <a href="#">View Tutorial</a>   <a href="#">Help</a>                               |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| <b>Select Search Type and Enter Search Terms</b>  |   |  |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| <div> <div>Terms &amp; Connectors</div> <div>Natural Language</div> <div>Easy Search™</div> </div>  | <div>7930576 or 7,930,576</div> <div></div> | <div>Suggest terms for my search</div> <div>Search</div> <div>Check spelling</div> |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| <b>Restrict by Document Segment</b>   |   |  |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| Select a document segment, enter search terms for the segment, then click Add.  |   |  |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| <div>Select a Segment</div> <div></div>   | <div>Add</div>                              |  |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| <b>Note:</b> Segment availability differs between sources. Segments may not be applied consistently across sources.   |   |  |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| <b>Restrict by Date</b>   |   |  |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| <div> <input checked="" type="radio"/> No Date Restrictions           <input type="radio"/> From <div></div> To <div></div> <a href="#">Date formats...</a> </div>  |   |  |                   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| <b>Search Connectors</b><br><table> <tr> <td>and</td> <td>and</td> <td>w/p</td> <td>in same paragraph</td> </tr> <tr> <td>or</td> <td>or</td> <td>w/seg</td> <td>in same segment</td> </tr> <tr> <td>w/N</td> <td>within N words</td> <td>w/s</td> <td>in same sentence</td> </tr> <tr> <td>pre/N</td> <td>precedes by N words</td> <td>and not</td> <td>and not</td> </tr> </table> <div>More Connectors &amp; Commands...</div> |   | and  | and               | w/p | in same paragraph | or | or | w/seg | in same segment | w/N | within N words | w/s | in same sentence | pre/N | precedes by N words | and not | and not | <b>How Do I...?</b><br>Combine sources?<br>Restrict by date?<br>Restrict by document segment?<br>Use wildcards as placeholders for one or more characters in a search term?<br><a href="#">View Tutorials</a> |
| and   | and   | w/p  | in same paragraph |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| or  | or  | w/seg  | in same segment   |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| w/N   | within N words                              | w/s  | in same sentence  |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |
| pre/N   | precedes by N words                         | and not  | and not           |     |                   |    |    |       |                 |     |                |     |                  |       |                     |         |         |   |

In

[About LexisNexis](#) | [Privacy Policy](#) | [Terms & Conditions](#) | [Contact Us](#)  
 Copyright © 2011 LexisNexis, a division of Reed Elsevier Inc. All rights reserved.



| No Documents Found  |
|---|
| <p>No documents were found for your search terms<br/><b>"7930576 or 7,930,576"</b></p> <p>Click "Save this search as an Alert" to schedule your search to run in the future.</p> <p>- OR -</p> <p>Click "Edit Search" to return to the search form and modify your search.</p> <p>Suggestions:</p> <ul style="list-style-type: none"><li>• Check for spelling errors.</li><li>• Remove some search terms.</li><li>• Use more common search terms, such as those listed in "Suggested Words and Concepts."</li><li>• Use a less restrictive date range.</li></ul> <p><a href="#">  Save this Search as an Alert</a>    <a href="#">Edit Search</a>  </p> |






In

[About LexisNexis](#) | [Privacy Policy](#) | [Terms & Conditions](#) | [Contact Us](#)  
Copyright © 2011 LexisNexis, a division of Reed Elsevier Inc. All rights reserved.

[Switch Client](#) | [Preferences](#) | [Help](#) | [Sign Out](#)

|                  |               |                            |                       |                      |                  |               |
|------------------|---------------|----------------------------|-----------------------|----------------------|------------------|---------------|
| <b>My Lexis™</b> | <b>Search</b> | <b>Get a Document</b>      | <b>Shepard's®</b>     | <b>More</b>          | <b>History</b>   | <b>Alerts</b> |
| <b>All</b>       | <b>Legal</b>  | <b>News &amp; Business</b> | <b>Public Records</b> | <b>Find A Source</b> | Add/Edit Subtabs |               |

Command Searching > Patent, Trademark & Copyright Periodicals, Combined 

|   |                      |   |
|---|----------------------|---|
| <b>Search</b>   |                      | <a href="#">View Tutorial</a>   <a href="#">Help</a>  |
| <b>Select Search Type and Enter Search Terms</b>  |                      |   |
| <b>Terms &amp; Connectors</b>   | 7930576 or 7,930,576 |              |
| Natural Language  |                      |   |
| Easy Search™  |                      |   |
|   |                      |              |
|   |                      | <a href="#">Suggest terms for my search</a>   |
|   |                      | <a href="#">Check spelling</a>  |
| <b>Search</b>   |                      |   |
| <b>Restrict by Document Segment</b>   |                      |   |
| Select a document segment, enter search terms for the segment, then click Add.  |                      |   |
| Select a Segment   |                      | <b>Add</b>  |
| <b>Note:</b> Segment availability differs between sources. Segments may not be applied consistently across sources.                             |                      |   |
| <b>Restrict by Date</b>   |                      |   |
| <input checked="" type="radio"/> <b>No Date Restrictions</b>  |                      |   |
| <input type="radio"/> From <input type="text"/> To <input type="text"/> <a href="#">Date formats...</a>   |                      |   |
| <b>Search Connectors</b>  |                      | <b>How Do I...?</b>   |
| and   | and                  | Combine sources?  |
| or  | or                   | Restrict by date?   |
| w/N   | within N words       | Restrict by document segment?   |
| pre/N   | precedes by N words  | Use wildcards as placeholders for one or more characters in a search term?                    |
|   | and not              | <a href="#">View Tutorials</a>  |
|   | and not              |   |
| <a href="#">More Connectors &amp; Commands...</a>   |                      |   |

In

[About LexisNexis](#) | [Privacy Policy](#) | [Terms & Conditions](#) | [Contact Us](#)  
Copyright © 2011 LexisNexis, a division of Reed Elsevier Inc. All rights reserved.

| No Documents Found  |
|---|
| <p>No documents were found for your search terms<br/><b>"7930576 or 7,930,576"</b></p> <p>Click "Save this search as an Alert" to schedule your search to run in the future.</p> <p>- OR -</p> <p>Click "Edit Search" to return to the search form and modify your search.</p> <p>Suggestions:</p> <ul style="list-style-type: none"><li>• Check for spelling errors.</li><li>• Remove some search terms.</li><li>• Use more common search terms, such as those listed in "Suggested Words and Concepts."</li><li>• Use a less restrictive date range.</li></ul> <p>  <a href="#">Save this Search as an Alert</a>    <a href="#">Edit Search</a>  </p> |

In

[About LexisNexis](#) | [Privacy Policy](#) | [Terms & Conditions](#) | [Contact Us](#)  
Copyright © 2011 LexisNexis, a division of Reed Elsevier Inc. All rights reserved.

[Switch Client](#) | [Preferences](#) | [Help](#) | [Sign Out](#)

|                  |               |                            |                       |                      |                  |               |
|------------------|---------------|----------------------------|-----------------------|----------------------|------------------|---------------|
| <b>My Lexis™</b> | <b>Search</b> | <b>Get a Document</b>      | <b>Shepard's®</b>     | <b>More</b>          | <b>History</b>   | <b>Alerts</b> |
| <b>All</b>       | <b>Legal</b>  | <b>News &amp; Business</b> | <b>Public Records</b> | <b>Find A Source</b> | Add/Edit Subtabs |               |

Command Searching &gt; News, All (English, Full Text) ⓘ

|   |                            |  |
|---|----------------------------|--|
| <b>Search</b>   |                            | <a href="#">View Tutorial</a>   <a href="#">Help</a>                       |
| <b>Select Search Type and Enter Search Terms</b>  |                            |  |
| <b>Terms &amp; Connectors</b>   | 7930576 or 7,930,576       | <input type="button" value="Search"/>                                      |
| <input type="button" value="Natural Language"/>   |                            | <input type="button" value="Suggest terms for my search"/>                 |
| <input type="button" value="Easy Search™"/>   |                            | <input type="button" value="Check spelling"/>                              |
| <b>Restrict by Document Segment</b>   |                            |  |
| Select a document segment, enter search terms for the segment, then click Add.                                      |                            |  |
| <input type="button" value="Select a Segment"/>   |                            | <input type="button" value="Add ^"/>                                       |
| <b>Note:</b> Segment availability differs between sources. Segments may not be applied consistently across sources. |                            |  |
| <b>Restrict by Date</b>   |                            |  |
| <input checked="" type="radio"/> <input type="button" value="No Date Restrictions"/>                                | <input type="radio"/> From | To <input type="text"/> <a href="#">Date formats...</a>                    |
| <b>Search Connectors</b>  |                            | <b>How Do I...?</b>  |
| and   | and                        | Combine sources?   |
| or  | or                         | Restrict by date?  |
| w/N   | within N words             | Restrict by document segment?  |
| pre/N   | precedes by N words        | Use wildcards as placeholders for one or more characters in a search term? |
|   | w/p                        | <a href="#">View Tutorials</a>   |
|   | w/seg                      |  |
|   | w/s                        |  |
|   | and not                    |  |
|   | and not                    |  |
| <a href="#">More Connectors &amp; Commands...</a>   |                            |  |

In

[About LexisNexis](#) | [Privacy Policy](#) | [Terms & Conditions](#) | [Contact Us](#)  
Copyright © 2011 LexisNexis, a division of Reed Elsevier Inc. All rights reserved.

Switch Client | Preferences | Help | Sign Out

|                  |               |                       |                   |             |                |
|------------------|---------------|-----------------------|-------------------|-------------|----------------|
| <b>My Lexis™</b> | <b>Search</b> | <b>Get a Document</b> | <b>Shepard's®</b> | <b>More</b> | <b>History</b> |
|                  |               |                       |                   |             | <b>Alerts</b>  |

FOCUS™ Terms Search Within View  
Tutorial

Advanced...

Source: **Command Searching > News, All (English, Full Text) (i)**Terms: **7930576 or 7,930,576** (Suggest Terms for My Search)

Select for FOCUS™ or Delivery

- ☐ 1. Patent Law Practice Center, June 29, 2011 Wednesday 5:00 AM EST, , 733 words, Challenge To Unigenes Osteoporosis Patent Among The Reexamination Requests Week Of 6/20/11, Stefanie Levine
- CORE TERMS:** patent, entitled, electronically, METHOD, APPARATUS, reexamination, litigation, styled, inter partes, PUMP, Hydro-Quebec, Unigene, Apotex, LLC, paper filed, Automation, Brookwood, Daritech, Newstex, Muth, DECODING, S.D.N.Y, IMAGING, CODING, PTO, formulation, requested, lithium, battery
- ... 2011. (5) 90/011,754 (electronically filed) " U.S. Patent No. **7,930,576** entitled SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A ...
- ☐ 2. Targeted News Service, April 22, 2011 Friday 2:03 PM EST, , 9684 words, U.S. Patents Awarded to Inventors in Texas (April 22), Targeted News Service Targeted News Service, Alexandria, VA.
- CORE TERMS:** patent, assigned, ISD, International Business Machines, published, http, nph-Parser, Sect1, Sect2, HITOFF, full-text, abstract, netacgi, patft, uspto, gov, Trademark Office, Austin, co-inventors, method, PTO2, PTXT, ASST, co1, Panigrahi, Satyaban, Hemanta, Armonk, Rath, data processing
- ... Va., April 22 -- Standard Microsystems, Hauppauge, N.Y., has been assigned a patent (**7,930,576**) developed by Ian F. Harris, Kings Park, N.Y., and Drew J. ...
- ☐ 3. Targeted News Service, April 22, 2011 Friday 10:14 AM EST, , 324 words, Standard Microsystems Assigned Patent, Targeted News Service, Alexandria, Va.
- CORE TERMS:** microcontroller, non-volatile, patent, reset, Myron, ISD, instructions, sharing, switches
- ... Va., April 22 -- Standard Microsystems, Hauppauge, N.Y., has been assigned a patent (**7,930,576**) developed by Ian F. Harris, Kings Park, N.Y., and Drew J. ...
- ☐ 4. Targeted News Service, April 22, 2011 Friday 10:14 AM EST, , 324 words, Standard Microsystems Assigned Patent, Targeted News Service, Alexandria, Va.
- CORE TERMS:** microcontroller, non-volatile, patent, reset, Myron, ISD, instructions, sharing, switches
- ... Va., April 22 -- Standard Microsystems, Hauppauge, N.Y., has been assigned a patent (**7,930,576**) developed by Ian F. Harris, Kings Park, N.Y., and Drew J. ...

- ☐ 5. US Fed News, April 20, 2011 Wednesday 10:23 AM EST, , 300 words, US Patent Issued to Standard Microsystems on April 19 for "Sharing Non-Sharable Devices Between an Embedded Controller and a Processor in a Computer System" (Texas, New York Inventors), ALEXANDRIA, Va.

**CORE TERMS:** microcontroller, non-volatile, patent, reset, instructions, sharing, switches, please

ALEXANDRIA, Va., April 20 -- United States Patent no. **7,930,576**, issued on April 19, was assigned to Standard Microsystems Corp. (Hauppauge, N.Y.). " ...  
... r=1&f=G&l=50&co1=AND&d=PTXT&s1=**7930576**&OS=**7930576**&RS=**7930576**  
For any query with respect to this article or any other content requirement, please  
...

Source: **Command Searching > News, All (English, Full Text)** 

Terms: **7930576 or 7,930,576** (Suggest Terms for My Search)

View: Cite

Date/Time: Friday, August 19, 2011 - 11:03 AM EDT

In

About LexisNexis | Privacy Policy | Terms & Conditions | Contact Us  
Copyright © 2011 LexisNexis, a division of Reed Elsevier Inc. All rights reserved.



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| REEXAM CONTROL NUMBER | FILING OR 371 (c) DATE | PATENT NUMBER |
|-----------------------|------------------------|---------------|
| 90/011,754            | 06/21/2011             | 7930576       |

JIAWEI HUANG (J.C. PATENTS)  
4, VENTURE  
SUITE 250  
IRVINE, CA 92618

**CONFIRMATION NO. 4071  
REEXAMINATION REQUEST  
NOTICE**



Date Mailed: 06/27/2011

**NOTICE OF REEXAMINATION REQUEST FILING DATE**

***(Third Party Requester)***

Requester is hereby notified that the filing date of the request for reexamination is 06/21/2011, the date that the filing requirements of 37 CFR § 1.510 were received.

A decision on the request for reexamination will be mailed within three months from the filing date of the request for reexamination. (See 37 CFR 1.515(a)).

A copy of the Notice is being sent to the person identified by the requester as the patent owner. Further patent owner correspondence will be the latest attorney or agent of record in the patent file. (See 37 CFR 1.33). Any paper filed should include a reference to the present request for reexamination (by Reexamination Control Number).

cc: Patent Owner

35690

MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.

P.O. BOX 398

AUSTIN, TX 78767-0398

/jawhitfield/

Legal Instruments Examiner

Central Reexamination Unit 571-272-7705; FAX No. 571-273-9900



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| REEXAM CONTROL NUMBER | FILING OR 371 (c) DATE | PATENT NUMBER |
|-----------------------|------------------------|---------------|
| 90/011,754            | 06/21/2011             | 7930576       |

35690  
MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.  
P.O. BOX 398  
AUSTIN, TX 78767-0398

**CONFIRMATION NO. 4071**  
**REEXAM ASSIGNMENT NOTICE**



Date Mailed: 06/27/2011

**NOTICE OF ASSIGNMENT OF REEXAMINATION REQUEST**

The above-identified request for reexamination has been assigned to Art Unit 3992. All future correspondence to the proceeding should be identified by the control number listed above and directed to the assigned Art Unit.

A copy of this Notice is being sent to the latest attorney or agent of record in the patent file or to all owners of record. (See 37 CFR 1.33(c)). If the addressee is not, or does not represent, the current owner, he or she is required to forward all communications regarding this proceeding to the current owner(s). An attorney or agent receiving this communication who does not represent the current owner(s) may wish to seek to withdraw pursuant to 37 CFR 1.36 in order to avoid receiving future communications. If the address of the current owner(s) is unknown, this communication should be returned within the request to withdraw pursuant to Section 1.36.

cc: Third Party Requester(if any)  
JIAWEI HUANG (J.C. PATENTS)  
4, VENTURE  
SUITE 250  
IRVINE, CA 92618

/jawhitfield/

Legal Instruments Examiner  
Central Reexamination Unit 571-272-7705; FAX No. 571-273-9900





## UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
 United States Patent and Trademark Office  
 Address: COMMISSIONER FOR PATENTS  
 P.O. Box 1450  
 Alexandria, Virginia 22313-1450  
 www.uspto.gov



Bib Data Sheet

CONFIRMATION NO. 4071

|   |   |                         |   |   |
|---|---|-------------------------|---|---|
| <b>SERIAL NUMBER</b><br>90/011,754  | <b>FILING OR 371(c) DATE</b><br>06/21/2011<br><b>RULE</b>   | <b>CLASS</b><br>713     | <b>GROUP ART UNIT</b><br>3992   | <b>ATTORNEY DOCKET NO.</b><br>JCLA38862-RE                      |
| <b>APPLICANTS</b><br>7,930,576 B2, Residence Not Provided;<br>STANDARD MICROSYSTEMS CORPORATION (OWNER), HAUPPAUGE, NY;<br>FELIX YEN (3RD PTY. REQ.), TAIPEI, TAIWAN;<br>JIAWEI HUANG (J.C. PATENTS), IRVINE, CA  |   |                         |   |   |
| <b>** CONTINUING DATA *****</b><br>This application is a REX of 11/958,601 12/18/2007 PAT 7,930,576<br>which claims benefit of 60/910,863 04/10/2007  |   |                         |   |   |
| <b>** FOREIGN APPLICATIONS *****</b>  |   |                         |   |   |
| Foreign Priority claimed <input type="checkbox"/> yes <input type="checkbox"/> no<br>35 USC 119 (a-d) conditions <input type="checkbox"/> yes <input type="checkbox"/> no <input type="checkbox"/> Met after<br>met Allowance<br>Verified and Acknowledged _____<br>Examiner's Signature Initials |   | <b>STATE OR COUNTRY</b> | <b>SHEETS DRAWING</b>   | <b>TOTAL CLAIMS</b><br>23<br><br><b>INDEPENDENT CLAIMS</b><br>4 |
| <b>ADDRESS</b><br>35690   |   |                         |   |   |
| <b>TITLE</b><br>SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM  |   |                         |   |   |
| <b>FILING FEE RECEIVED</b><br>2520  | FEES: Authority has been given in Paper<br>No. _____ to charge/credit DEPOSIT ACCOUNT<br>No. _____ for following: |                         | <input type="checkbox"/> All Fees<br><input type="checkbox"/> 1.16 Fees ( Filing )<br><input type="checkbox"/> 1.17 Fees ( Processing Ext. of time )<br><input type="checkbox"/> 1.18 Fees ( Issue )<br><input type="checkbox"/> Other _____<br><input type="checkbox"/> Credit |   |

(Also referred to as FORM PTO-1465)

**REQUEST FOR *EX PARTE* REEXAMINATION TRANSMITTAL FORM**

Address to:

**Mail Stop *Ex Parte* Reexam  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450**

Attorney Docket No.: JCLA38862-RE

Date: JUNE 21, 2011

1. ☒ This is a request for *ex parte* reexamination pursuant to 37 CFR 1.510 of patent number 7,930,576 issued April 19, 2011. The request is made by:  
☐ patent owner. ☒ third party requester.
2. ☒ The name and address of the person requesting reexamination is:  
Felix Yeh  
7th Floor-1, No. 100 Roosevelt Road, Section 2  
Taipei, Taiwan
3. ☐ a. A check in the amount of \$\_\_\_\_\_ is enclosed to cover the reexamination fee, 37 CFR 1.20(c)(1);  
☒ b. The Director is hereby authorized to charge the fee as set forth in 37 CFR 1.20(c)(1) to Deposit Account No. 50-0710; or  
☐ c. Payment by credit card. Form PTO-2038 is attached.
4. ☒ Any refund should be made by ☐ check or ☒ credit to Deposit Account No. 50-0710 37 CFR 1.26(c). If payment is made by credit card, refund must be to credit card account.
5. ☒ A copy of the patent to be reexamined having a double column format on one side of a separate paper is enclosed. 37 CFR 1.510(b)(4)
6. ☐ CD-ROM or CD-R in duplicate, Computer Program (Appendix) or large table  
☐ Landscape Table on CD
7. ☐ Nucleotide and/or Amino Acid Sequence Submission  
*If applicable, items a. – c. are required.*  
a. ☐ Computer Readable Form (CRF)  
b. Specification Sequence Listing on:  
i. ☐ CD-ROM (2 copies) or CD-R (2 copies); or  
ii. ☐ paper  
c. ☐ Statements verifying identity of above copies
8. ☐ A copy of any disclaimer, certificate of correction or reexamination certificate issued in the patent is included.
9. ☒ Reexamination of claim(s) 1-23 is requested.
10. ☒ A copy of every patent or printed publication relied upon is submitted herewith including a listing thereof on Form PTO/SB/08, PTO-1449, or equivalent.
11. ☐ An English language translation of all necessary and pertinent non-English language patents and/or printed publications is included.

[Page 1 of 2]

This collection of information is required by 37 CFR 1.510. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 18 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop *Ex Parte* Reexam, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

*If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.*

|  |                |   |                                      |
|--|----------------|---|--------------------------------------|
| 12. <input checked="" type="checkbox"/> The attached detailed request includes at least the following items: <div style="margin-left: 20px;">           a. A statement identifying each substantial new question of patentability based on prior patents and printed publications. 37 CFR 1.510(b)(1)<br/>           b. An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited art to every claim for which reexamination is requested. 37 CFR 1.510(b)(2).         </div>   |                |   |                                      |
| 13. <input type="checkbox"/> A proposed amendment is included (only where the patent owner is the requester). 37 CFR 1.510(e)  |                |   |                                      |
| 14. <input checked="" type="checkbox"/> a. It is certified that a copy of this request (if filed by other than the patent owner) has been served in its entirety on the patent owner as provided in 37 CFR 1.33(c).<br>The name and address of the party served and the date of service are:<br><u>MEYERTONS, HOOD, KIVLIN, KOWERT &amp; GOETZEL, P.C.</u><br><u>P.O. BOX 398</u><br><u>AUSTIN, TX 78767-0398</u><br><br>Date of Service: <u>JUNE 21, 2011</u> ; or<br><input type="checkbox"/> b. A duplicate copy is enclosed because service on patent owner was not possible. An explanation of the efforts made to serve patent owner <b>is attached</b> . See MPEP 2220. |                |   |                                      |
| 15. Correspondence Address: Direct all communications about the reexamination to: <div style="margin-left: 20px;"> <input type="checkbox"/> The address associated with Customer Number: <span style="border: 1px solid black; display: inline-block; width: 200px; height: 20px; vertical-align: middle;"></span><br/> <b>OR</b><br/> <input checked="" type="checkbox"/> Firm or Individual Name <u>JIAWEI HUANG (J. C. Patents)</u> </div>  |                |   |                                      |
| Address<br><div style="margin-left: 40px;">4, Venture Suite 250</div>  |                |   |                                      |
| City   | Irvine         | State   | CA                                   |
| Zip  | 92618          |   |                                      |
| Country  | U.S.A.         |   |                                      |
| Telephone  | (949) 660-0761 |   | Email <u>jcpatents@sbcglobal.net</u> |
| 16. <input type="checkbox"/> The patent is currently the subject of the following concurrent proceeding(s): <div style="margin-left: 20px;"> <input type="checkbox"/> a. Copending reissue Application No. _____<br/> <input type="checkbox"/> b. Copending reexamination Control No. _____<br/> <input type="checkbox"/> c. Copending Interference No. _____<br/> <input type="checkbox"/> d. Copending litigation styled: _____<br/>           _____<br/>           _____         </div>   |                |   |                                      |
| <b>WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.</b>  |                |   |                                      |
| <u>/JIAWEI HUANG/</u><br>Authorized Signature  |                | <u>June 21, 2011</u><br>Date  |                                      |
| <u>JIAWEI HUANG</u><br>Typed/Printed Name  |                | <u>43,330</u> <input type="checkbox"/> For Patent Owner Requester<br>Registration No. <input checked="" type="checkbox"/> For Third Party Requester |                                      |

## Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.



US007930576B2

(12) **United States Patent**  
**Harris et al.**

(10) **Patent No.:** **US 7,930,576 B2**  
(45) **Date of Patent:** **Apr. 19, 2011**

(54) **SHARING NON-SHARABLE DEVICES  
BETWEEN AN EMBEDDED CONTROLLER  
AND A PROCESSOR IN A COMPUTER  
SYSTEM**

(75) Inventors: **Ian F. Harris**, Kings Park, NY (US);  
**Drew J. Dutton**, Austin, TX (US)

(73) Assignee: **Standard Microsystems Corporation**,  
Hauppauge, NY (US)

(\*) Notice: Subject to any disclaimer, the term of this  
patent is extended or adjusted under 35  
U.S.C. 154(b) by 684 days.

(21) Appl. No.: **11/958,601**

(22) Filed: **Dec. 18, 2007**

(65) **Prior Publication Data**  
US 2008/0256374 A1 Oct. 16, 2008

**Related U.S. Application Data**  
(60) Provisional application No. 60/910,863, filed on Apr.  
10, 2007.

(51) **Int. Cl.**  
**G06F 1/26** (2006.01)  
**G06F 15/177** (2006.01)  
(52) **U.S. Cl.** ..... **713/324**; 713/2; 713/100  
(58) **Field of Classification Search** ..... 713/324,  
713/2, 100

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,428,760 A \* 6/1995 Ghorl et al. .... 711/149  
5,634,004 A \* 5/1997 Gopinath et al.  
5,819,087 A \* 10/1998 Le et al. .... 713/2  
5,925,097 A \* 7/1999 Gopinath et al.  
5,949,997 A \* 9/1999 Smith ..... 713/2  
6,154,838 A \* 11/2000 Le et al. .... 713/2  
6,925,522 B2 \* 8/2005 Pang ..... 711/103  
7,386,326 B2 \* 6/2008 Sundararajan et al. .... 455/561

\* cited by examiner

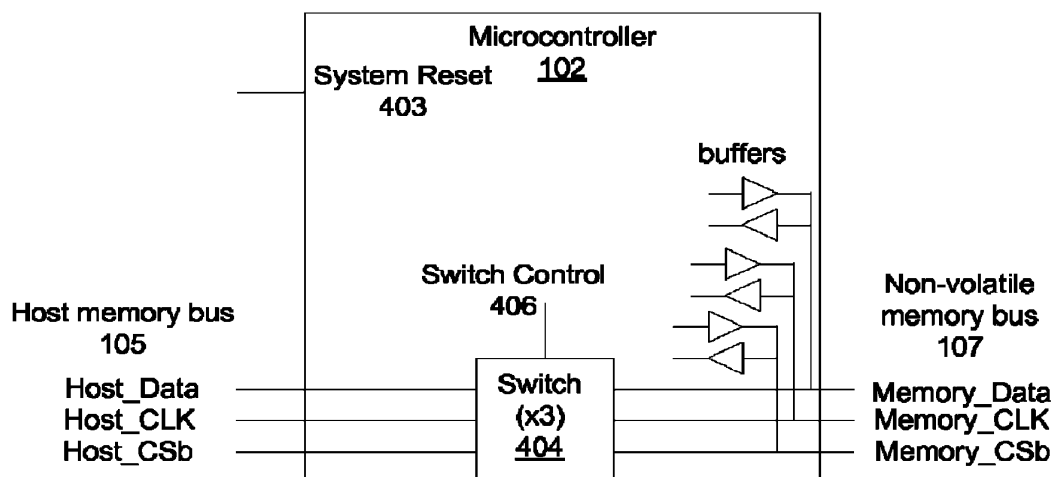
*Primary Examiner* — Nitin C Patel

(74) *Attorney, Agent, or Firm* — Meyertons Hood Kivlin  
Kowert & Goetzel, P.C.; Jeffrey C. Hood; Mark S. Williams

(57) **ABSTRACT**

System and method for sharing a device, e.g., non-volatile  
memory, between a host processor and a microcontroller. In  
response to system state change to a first state wherein the  
microcontroller is assured safe access to the non-volatile  
memory (e.g., in response to power-on reset, system reset,  
sleep state, etc.), the microcontroller holds the system in the  
first state (e.g., system reset), and switches access to the  
non-volatile memory from the processor to the microcontrol-  
ler. While the system is held in the first state, the microcon-  
troller accesses the device (e.g., non-volatile memory), e.g.,  
fetches program instructions/data from the non-volatile  
memory and loads the program instructions/data into a  
memory of the microcontroller. After the access, the micro-  
controller changes or allows change of the system state, e.g.,  
switches access to the device, e.g., the non-volatile memory,  
from the microcontroller to the processor, and releases the  
system from the first state.

**23 Claims, 5 Drawing Sheets**



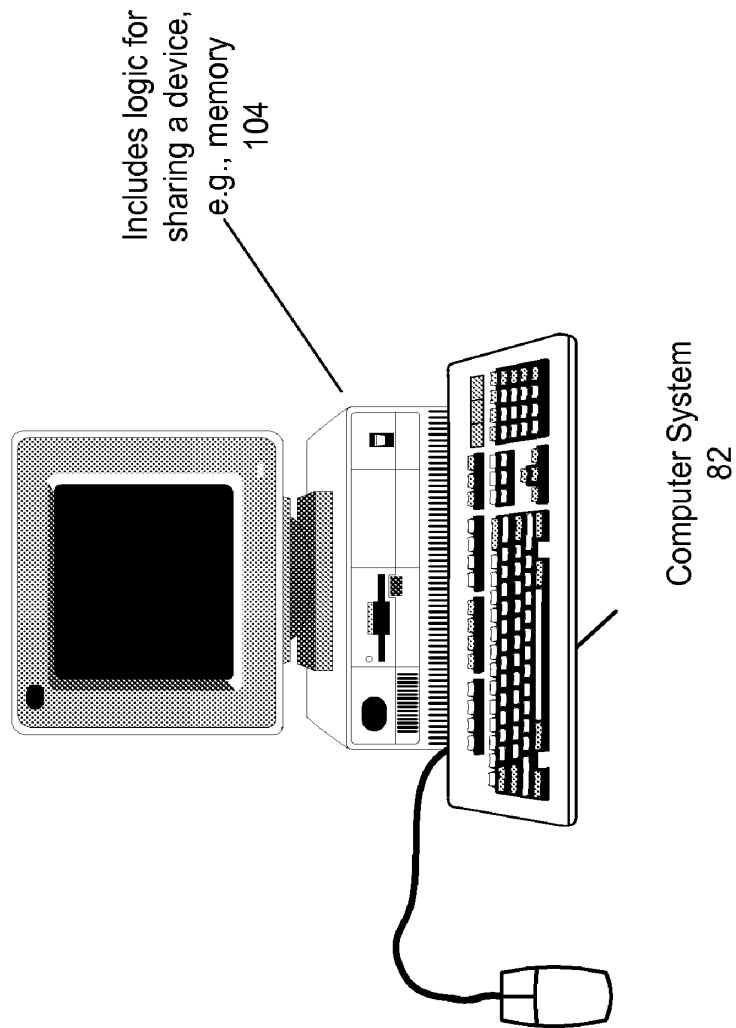


Figure 1A

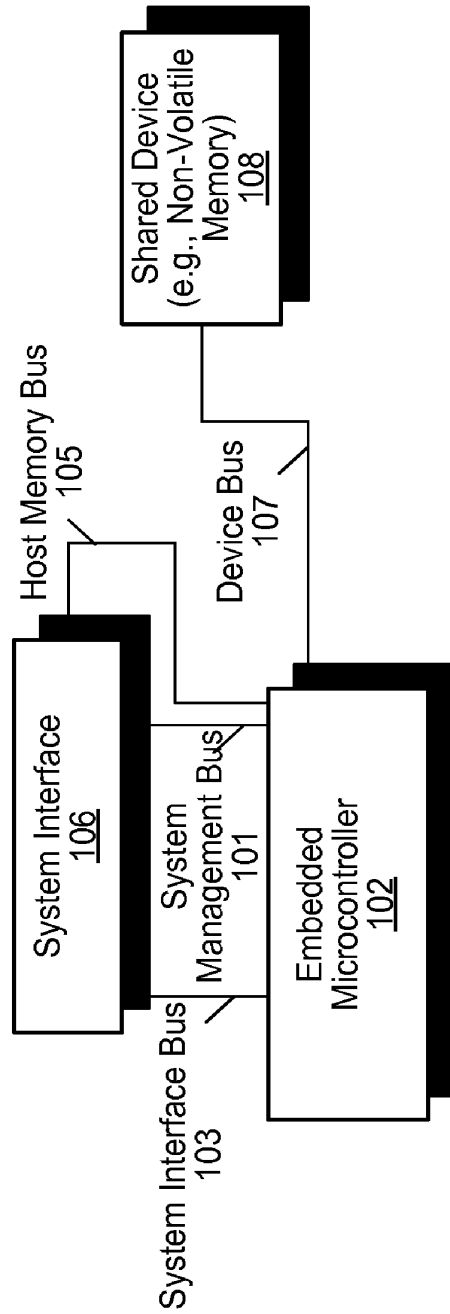


Figure 1B

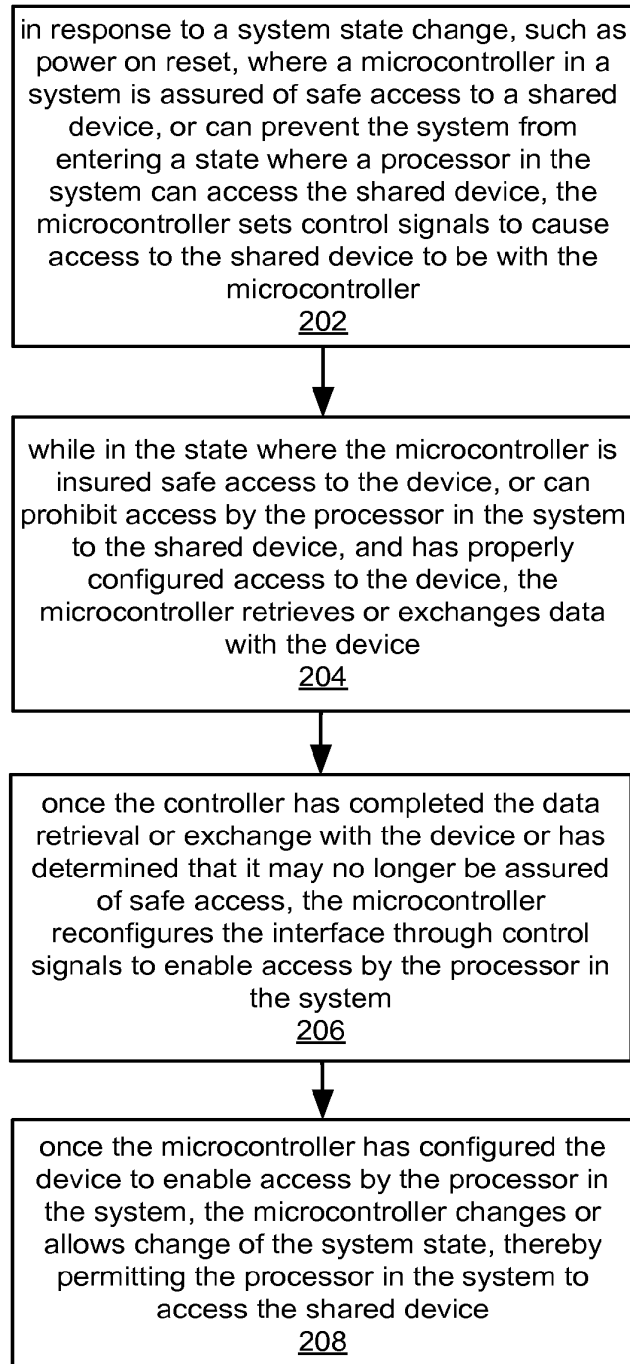


Figure 2



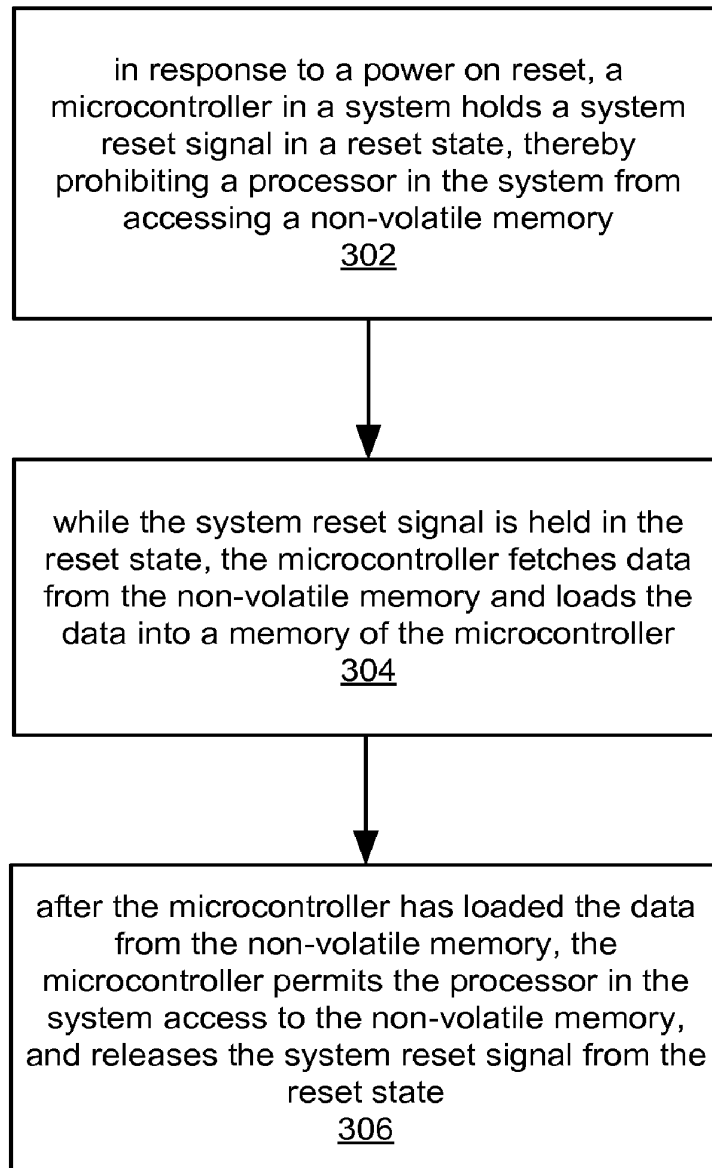


Figure 3

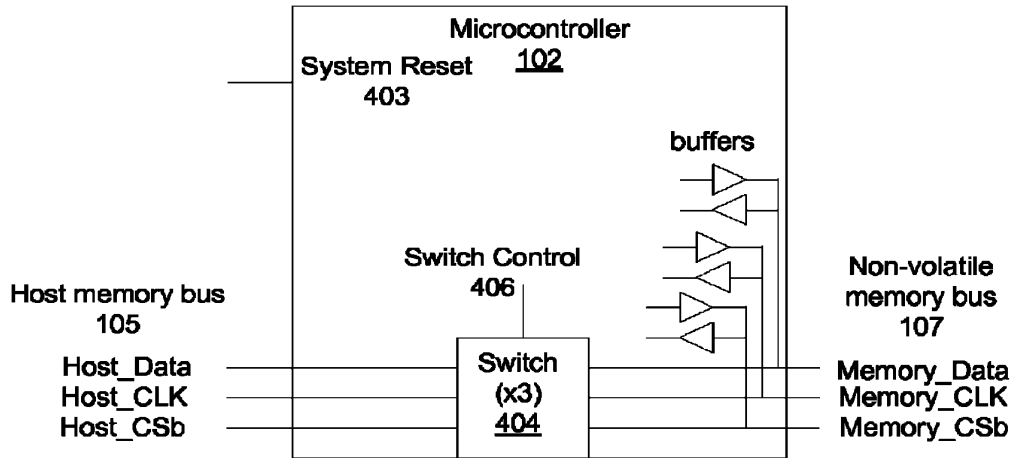


Figure 4

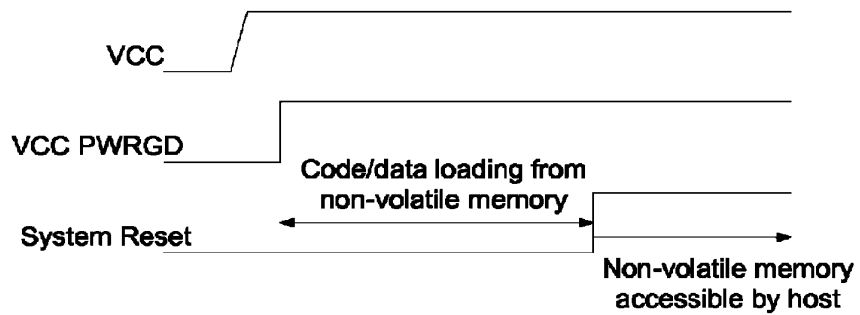


Figure 5

1

**SHARING NON-SHARABLE DEVICES  
BETWEEN AN EMBEDDED CONTROLLER  
AND A PROCESSOR IN A COMPUTER  
SYSTEM**

PRIORITY DATA

This application claims benefit of priority to U.S. Provisional Application Ser. No. 60/910,863, titled "Sharing Non-Sharable Devices Between an Embedded Controller and A Processor in a Computer System", filed on Apr. 10, 2007, whose inventors were Ian Harris and Drew J. Dutton.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to the field of computing devices, and more specifically to sharing devices, such as non-volatile memory, which are not designed for sharing among processors in a system.

2. Description of the Related Art

Many hardware systems include a central processing unit (CPU), i.e., a main processor, and one or more microcontrollers for performing auxiliary functions for the system, such as initialization, management and reset functionality. Each processor or microcontroller, in a system generally has a corresponding non-volatile memory from which the processor or microcontroller reads instructions for performing boot-up, reset, management (such as power state management), or other functions, implemented in firmware. Current solutions for providing controller memory are: (1) to have a separate dedicated/private memory device for each controller/processor; (2) to integrate memory on-chip; or (3) to attempt to arbitrate for the use of an existing memory device.

However, a problem with attempting to arbitrate for the use of another device is that system performance and user experience may be degraded. Moreover, providing each processor and microcontroller with a separate non-volatile memory is expensive in both cost and time. For example, many current embedded microcontroller solutions for notebook and desktop computers use ROM (read-only memory) solutions that present challenges in time to market and flexibility. The time to market issue is due to the need to finalize the microcontroller firmware and data early enough in the design and manufacturing process to meet the schedule needs for production. This usually represents at least a month in the time schedule of a development program. In addition, any late error found in the firmware or data means that all existing inventory is out of date and may have to be discarded.

For the above reasons, some customers or designers prefer using embedded flash or similar memory for storage, since this type of memory allows updates at production test or even in the field. However, use of embedded flash requires a very expensive manufacturing process, because the microcontroller chip integrates flash memory on the same die, and therefore requires a flash manufacturing process. It is typically 30% more expensive to manufacture a chip with this process, and thus makes the chip significantly more expensive. This expense only makes sense if offset by some other factors, e.g., cost savings or product requirements. If a customer only has one set of firmware (code) that is used and if the code never changes, and if the customer is confident that there are no errors in the code that might require an update, ROM and a standard manufacturing process may be used instead of embedded flash memory. If, on the other hand, the customer has multiple firmware codes (e.g., one per product model), then it may be cheaper to use flash. Thus, due to expense, use

2

of embedded flash memory may only be viable if the risk of ROM is too high or if there are multiple codes, low volume or other specific business reasons.

Thus, as discussed above, current solutions are generally expensive, inflexible, cannot be guaranteed to work reliably, may have occasional errors, and/or can cause significant system performance degradation.

Other corresponding issues related to the prior art will become apparent to one skilled in the art after comparing such prior art with embodiments of the present invention as described herein.

SUMMARY OF THE INVENTION

Various embodiments are presented of a system and method for sharing devices such as non-volatile memory between processors in a computer system where one of the processors has control over operating states, e.g., power and/or execution states, of the other processors in the computer system.

In response to a system state change, such as power on reset or system reset, where a microcontroller in a system is assured of safe access to a shared device, or can prevent the system from entering a state where a processor in the system can access the shared device, the microcontroller may set control signals to cause access to the shared device to be with the microcontroller. In other words, the system may be changed to a state (a "first" state) in which the microcontroller is guaranteed the ability to access the shared device safely, e.g., without interruption or interference by the processor.

In one embodiment, the microcontroller may be situated between a system interface (and therefore, the processor in the system) and the shared device, e.g., non-volatile memory, e.g., embedded flash memory, and thus may intercept communications between the system interface and the device. The device may include or be associated with an interface whereby access to the device is facilitated or implemented, and where this interface may be configured to direct or restrict access to the device to different system components, e.g., the processor or the microcontroller. In one embodiment, the microcontroller setting control signals to cause access to the shared device to be with the microcontroller comprises the microcontroller setting the interface to the shared device interface with the microcontroller. Said another way, the microcontroller may set one or more control signals to configure or redirect the shared device interface to grant exclusive access to the device to the microcontroller.

In one embodiment, the microcontroller controls and manages power on reset (POR) for the system, and thus may respond to and/or generate signals related to power on reset or system reset (or other management signals) in a way that preempts access to the device by the main processor of the system, thereby facilitating sharing a device that otherwise could not be shared. In some embodiments, the change in system state may be invoked or indicated by a management signal, e.g., a system reset or POR signal, generated or received by the microcontroller. It should be noted, however, that in some embodiments, control of the direction of the interface, i.e., control of which component of the system has access to the device, may be independent of the management (e.g., system reset or POR) signal. In other words, the microcontroller may control whether the shared device interface is directed to the processor of the system, or to the microcontroller, independent of the management (e.g., system reset or POR) signal.

While in the first state, i.e., the state where the microcontroller is ensured safe access to the device, or can prohibit

3

access by the processor in the system to the shared device, and has properly configured access to the device, the microcontroller may retrieve or exchange data with the device. For example, in embodiments where the system state is dependent on or indicated by a management signal, e.g., a system reset signal, the microcontroller may receive or generate the management signal, and hold the management signal in this state, i.e., may maintain assertion of the management signal, and, while this signal is asserted, may read, write, acquire, or otherwise exchange (e.g., via invocation of a read or write by some other component of the system), data or program instructions from, to, or with, the device.

Once the controller has completed the data retrieval or exchange with the device or has determined that it may no longer be assured of safe access, the microcontroller may reconfigure the interface through control signals to enable access by the processor in the system, e.g., via a switch that controls access to the device, as will be described below in more detail. Once the microcontroller has configured the device to enable access by the processor in the system, the microcontroller may change or allow change of the system state, thereby permitting the processor in the system to access the shared device. For example, in embodiments where the system state is dependent on or indicated by a management signal, e.g., a system reset signal, the microcontroller may de-assert the management signal, thereby putting the system into a state that allows access to the shared device by the processor of the system.

Thus, using embodiments of the above-described method, the microcontroller may preempt or suspend access to the device by the processor, e.g., the main CPU, of the system, or otherwise configure safe access to the device by the microcontroller; retrieve or otherwise exchange data or program instructions with the device; then return access to the device to the processor; or otherwise relinquish exclusive or safe access to the device by the microcontroller.

It should be noted that in various other embodiments where the microcontroller manages or controls operating states of the system (e.g., reset, POR, sleep, and other states related to specific system activities where it is known by the microcontroller that the system cannot or will not access the device), the microcontroller may monitor (or control) such system states, and take advantage of these states to access the device safely, e.g., to take control and use the shared device. Note that these system states should be such that the microcontroller either manages when the system will exit these states or is provided sufficient warning as to when the system will again need control of the shared interface (to the device). As one example, when a system is being shut down, the microcontroller may take over the device interface, i.e., may redirect the device interface to allow (only) access by the microcontroller, before power-off, i.e., before turning off the final power rail. In this way, writes or other exchanges with the shared device may be "saved up" or queued, and their completion (e.g., by the microcontroller) may be guaranteed before final power off.

Thus, the microcontroller may gracefully share access to the device with the processor of the system. It should be noted that in other embodiments, additional microcontrollers may also share the device with the host or main system CPU, using the techniques described above. In further embodiments, such sharing of devices may also be used in other ways. For example, in some embodiments, a plurality of processors (e.g., microcontrollers) may share such a device, where each processor may in turn suspend or prevent access to the device by the other processors while it accesses the device.

4

Below are described more specific embodiments of the above-described general method that are particularly directed to the sharing of non-volatile memory between a main processor and a microcontroller in a system. In other words, in the below embodiments, the shared device comprises a non-volatile memory device. It should be noted, however, that these embodiments are meant to be exemplary only, and are not intended to limit the type or use of the shared device to any particular form or function.

In response to a power on reset (POR), a microcontroller in a system, e.g., microcontroller, may hold a system reset signal in a reset state, thereby prohibiting a processor in the system from accessing a non-volatile memory in the system. Note that the POR and/or the system reset may or may not be invoked by the microcontroller.

In other words, in response to the POR, the microcontroller may receive or generate the system reset signal, and hold the signal in a reset state. As described above, in one embodiment, the microcontroller may be situated between the system interface (and therefore, the processor in the system) and the shared device, e.g., non-volatile memory, e.g., embedded flash memory, and thus may intercept communications between the system interface and the device, e.g., the non-volatile memory. As will be described below in more detail, in some embodiments, when the system reset signal is in the reset state, the interface to the non-volatile memory may be configured by the microcontroller to preclude the processor in the system from accessing the non-volatile memory, e.g., via a switch.

While the system reset signal (or other management signal) is held in the reset state, e.g., is asserted by the microcontroller (e.g., and the interface to the non-volatile memory is configured to only allow access by the microcontroller), the microcontroller may fetch program instructions/data from the non-volatile memory and load the program instructions/data into a memory of the microcontroller, or otherwise access the non-volatile memory. In other words, the microcontroller may generate or receive the system reset signal, and may hold the system reset signal in the reset state (e.g., on the system management bus via a system reset line), i.e., may maintain assertion of the system reset signal, and, while the system reset signal is asserted, may fetch, i.e., read, acquire, or otherwise receive (e.g., via invocation of a read by some other component of the system), program instructions/data from the non-volatile memory and load or write the program instructions/data into the microcontroller's memory. Note that the microcontroller's memory is preferably RAM (random access memory), although any type of re-writable memory may be used as desired. As noted above, in some embodiments, the program instructions/data may include program firmware code, that may, for example, be usable by the embedded microcontroller for startup or reset functionality, e.g., for initializing the microcontroller, among other uses. In some embodiments, the microcontroller may be or include a peripheral device controller for a computer system. For example, in one exemplary embodiment, the embedded microcontroller may be a keyboard controller, and the non-volatile memory device 108 may store keyboard controller (KBC) code, although in other embodiments, any other program instructions/code or data may be used as desired.

In one embodiment, the microcontroller may include or couple to a host memory bus, which may couple to the system interface, and may also include or couple to a device bus, in this embodiment, a non-volatile memory bus, which may couple to the shared non-volatile memory device. In one embodiment, the host memory bus and the non-volatile memory bus are SPI (serial peripheral interface) buses,

5

although other protocols may be used as desired. Note that in some embodiments, the host memory bus and the non-volatile memory bus may be considered the same bus, e.g., a memory bus, where the microcontroller is inserted to switchably intercept and possibly redirect signals on the bus(es).

In one embodiment, the microcontroller includes or is coupled to a switch controlled by a switch control, for providing switching functionality for the host memory bus and the non-volatile memory bus, e.g., for the memory bus. The microcontroller may thus be operable to switchably preempt and inhibit signal flow on the memory bus (i.e., the host memory bus and the non-volatile memory bus) between the host interface and the non-volatile memory, e.g., by placing the switches into a high-impedance state, and to fetch instructions/data from the non-volatile memory while communication between the host interface and the non-volatile memory is suspended. Thus, the microcontroller may operate to configure the interface to the device (e.g., the switch) to regulate and enforce safe access to the device.

After the microcontroller has loaded the program instructions/data from the non-volatile memory, the microcontroller may reconfigure the access to the non-volatile memory, e.g., via the switch, and release the system reset signal from the reset state, thereby permitting the processor in the system access to the non-volatile memory. In other words, upon completion of the fetch of instructions/data from the non-volatile memory by the microcontroller, the microcontroller may terminate the suspension or preemption of communications between host interface and the non-volatile memory, thereby allowing resumption of such communications, e.g., via switch, and de-assert the system reset signal.

Thus, using embodiments of the above-described method, the microcontroller may preempt or suspend communications with the non-volatile memory by the main CPU and read instructions/data from the non-volatile memory, e.g., to initialize the microcontroller and/or perform other functionality prior to, or independent from, operation of the main system, i.e., the main CPU, thus sharing the non-volatile memory with the main system. For example, in some embodiments, the microcontroller may, in addition to initialization code/data, may also fetch program instructions and/or data related to pre-boot security operations, and may operate to perform such pre-boot security operations prior to boot-up of the main system, thereby providing additional security to the system.

Thus, the microcontroller may gracefully share use of the main system's non-volatile memory. It should be noted that in other embodiments, additional microcontrollers may also share this non-volatile memory with the host or main system CPU, using the techniques described above. In further embodiments, such sharing of non-volatile memory may also be used in other ways. For example, in some embodiments, a plurality of microcontrollers may share a non-volatile memory (e.g., that is not used by the main system CPU), where each microcontroller may in turn suspend or prevent access to the non-volatile memory by the other microcontrollers while it loads instructions/data from the memory.

Note that in various embodiments, the system changing to a state where the microcontroller is assured safe access to the non-volatile memory (or device) may be performed or mediated by software or hardware. In other words, in some embodiments, the state change may be in response to hardware signals, such as POR signals, etc. In other embodiments, the state change may be in response to software events, tokens, software conditions, software messages, software handshakes, and so forth. Said another way, embodiments of the present invention are intended to cover virtually any case

6

where the embedded controller can determine that it can safely take control of the shared non-volatile memory, whether the "safe" system state is mediated or signaled via software or hardware. For example, the system may pass the microcontroller a token, which grants the microcontroller ownership of the memory through a software handshake.

Thus, various embodiments of the invention described herein may allow an embedded microcontroller and a processor in a system to share a device that is not designed to be shared.

#### BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing, as well as other objects, features, and advantages of this invention may be more completely understood by reference to the following detailed description when read together with the accompanying drawings in which:

FIG. 1A illustrates an exemplary computer system configured to implement one embodiment of the present invention;

FIG. 1B is a high-level block diagram of an exemplary system configured to implement one embodiment of the present invention;

FIG. 2 is a flowchart of a method for sharing a device between a processor and a microcontroller in a system, according to one embodiment;

FIG. 3 is a flowchart of a method for sharing non-volatile memory between a processor and a microcontroller in a system, according to one embodiment.

FIG. 4 illustrates a microcontroller, according to one embodiment of the invention; and

FIG. 5 illustrates an exemplary signal timing diagram, according to one embodiment.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

#### DETAILED DESCRIPTION OF THE EMBODIMENTS

##### Incorporation by Reference

The following references are hereby incorporated by reference in their entirety as though fully and completely set forth herein:

U.S. Provisional Application Ser. No. 60/910,863, titled "Sharing Non-Sharable Devices Between an Embedded Controller and A Processor in a Computer System", filed on Apr. 10, 2007, whose inventors were Ian Harris and Drew J. Dutton.

Various embodiments of a system and method are presented for sharing devices between processors, e.g., between a one or more main processors and a microcontroller, in a system, e.g., a computer system, where the microcontroller has control over the power or execution states of the main processor(s) or other unique management capabilities relative to the other processors in the system.

Note that any headings used are for organizational purposes only and are not meant to be used to limit or interpret the description or claims. Furthermore, it should be noted that the word "may" is used throughout this application in a permissive sense (i.e., having the potential to, being able to), not a

7

mandatory sense (i.e., must).” The term “include”, and derivations thereof, mean “including, but not limited to”. The term “coupled” means “directly or indirectly connected”.

Moreover, as used herein, a “set of instructions” may refer to one or more instructions. More specifically, in some embodiments, “instructions” may refer to programming code, software, and/or functions implemented in the form of a code that is executable by a controller, microprocessor, and/or custom logic circuit adapted to execute these instructions. In some embodiments, these instructions may comprise device drivers, control software, and/or machine code.

As used herein, a “controller” refers to any type of processor, such as a central processing unit (CPU), microprocessor, microcontroller, or embedded controller or microcontroller, among others.

#### Overview

Various embodiments of the invention described herein may provide access by an embedded microcontroller to a device (e.g., non-volatile memory) which allows the microcontroller to access and read or write program instructions/data (meaning program instructions and/or data) from or to the device while holding the system, e.g., the CPU, in a state such as reset where access to the non-volatile memory device will not disrupt the normal system function.

#### FIG. 1A—Exemplary Computer System

FIG. 1 is a high-level block diagram of an exemplary computer system **82** configured to implement some embodiments of the present invention. The system of FIG. 1 comprises a computer system, e.g., a personal computer (PC), although in other embodiments, the techniques and systems described herein may be implemented in any other systems as desired.

In preferred embodiments, the computer system **82** includes a main or central processor, i.e., a host processor, as well as one or more microcontrollers, e.g., each in a respective system-on-chip (SoC), such as, for example, a keyboard controller for managing or controlling communications between a keyboard and the main processor. The memory is preferably included on the SoC with the embedded controller, and in various embodiments, may be a non-volatile memory, e.g., ROM (read only memory), or a volatile memory, such as Flash memory, as desired. As FIG. 1 shows, the computer system **82** includes logic **104** for sharing memory, e.g., between the main processor and the microcontroller, according to embodiments of the present invention. Further details regarding exemplary embodiments of the system are provided below with reference to FIGS. 1B, 2, 4, and 6.

#### FIG. 1B—Exemplary System

FIG. 1B is a high-level block diagram of an exemplary system configured to implement one embodiment of the present invention, specifically to share a device, e.g., non-volatile memory, between a CPU or main processor, and an embedded microcontroller in the system. The system of FIG. 1B preferably resides in a computer system, e.g., a personal computer (PC) such as computer system **82** of FIG. 1A, although in other embodiments, the techniques and systems described herein may be implemented in any other systems as desired.

As FIG. 1B shows, the system may include an embedded microcontroller **102**, coupled to system interface **106** via a system interface bus **103**, whereby the microcontroller **102** may communicate with the CPU of the computer system, referred to as the host CPU or processor. Note that in various embodiments, this system interface **106** may simply be a connection or bus suitable for communications between the microcontroller **102** and the host CPU, or may include additional structure or functionality as desired. As indicated in FIG. 1B, the microcontroller **102** may be further coupled to a

8

shared device **108** that may store program instructions or other data that may be uploaded or programmed by the embedded microcontroller, e.g., for startup, general, or reset functionality, via a device bus **107**, as shown. The microcontroller **102** may be further coupled to the system interface **106** via a host memory bus **105**, whereby the embedded microcontroller **102** may intercept memory requests or other signals transmitted between the system interface (e.g., the host CPU) and the shared device **108**. Note that the program instructions/data stored in the shared device (e.g., non-volatile memory) **108** preferably includes program instructions, e.g., firmware code, that may, for example, be usable by the embedded microcontroller **102** for startup, general, or reset functionality, e.g., for initializing the microcontroller **102**, among other uses. In some embodiments, the microcontroller **102** may comprise a peripheral device controller for a computer system. For example, in one exemplary embodiment, the embedded microcontroller **102** is a keyboard controller, and the shared device **108** may store keyboard controller (KBC) code, although in other embodiments, any other program instructions, data or code may be used as desired.

Thus, in the embodiment shown in FIG. 1B, the embedded microcontroller **102** is situated between the system interface **106** and the shared device (e.g., non-volatile memory) **108**, and so may function as an intermediary or intercept between these two components, thereby facilitating sharing the device **108** between the host processor and the embedded microcontroller **102**, as will be described below in more detail. Note that in this embodiment, a system management bus, e.g., a system reset bus, **101** coupling the system interface **106** and the microcontroller **102** may be included for communicating a reset or other management signal (or other signal) between the two components, although in other embodiments, the system interface bus **103** may be used. In preferred embodiments, the shared device **108** is an embedded flash memory, although other types of devices may be used as desired, e.g., ROM or any other type of memory.

#### FIG. 2—Method for Sharing a Device Between Processors in a System

FIG. 2 is a high-level flowchart of embodiments of a method for sharing a device, e.g., non-volatile memory, between processors, e.g., between a processor and a microcontroller in a system, e.g., a computer system, where the shared device is not designed to be shared. In various embodiments, some of the method elements shown may be performed concurrently, in a different order than shown, or may be omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

In **202**, in response to a system state change, such as, for example, power on reset (POR), system reset, or sleep mode, where a microcontroller, e.g., microcontroller **102**, in a system is assured of safe access to a shared device, or can prevent the system from entering a state where a processor in the system can access the shared device, the microcontroller may set control signals or utilize software to cause access to the shared device to be with the microcontroller. In other words, when the system is changed to a state, e.g., a first state, in which the microcontroller **102** is guaranteed the ability to access the shared device safely, e.g., without interruption or interference by the processor, the microcontroller may hold the system in the state and switch access to the shared device, e.g., the non-volatile memory, from the processor to the microcontroller. Examples of this first state may include one or more of: a power on reset (POR) state, a system reset state, or a sleep state, among others. Holding the system in this first

state may prohibit the processor in the system from accessing the shared device, e.g., non-volatile memory in the system.

Similarly, the state change may be invoked or caused by a management signal, such as a POR, system reset, sleep, signal, etc., which may include a software signal, event, token, handshake, etc. In one embodiment, holding the system in the first state may include holding a management signal in a corresponding state, e.g., holding a system reset signal in a reset state.

Note that the management signal, e.g., the POR and/or the system reset, and/or any other type of management signal used, may or may not be invoked by the microcontroller. Thus, for example, in response to a received management signal, e.g., POR signal, or in response to generating the management signal, e.g., the POR signal, the microcontroller may generate or receive another management signal, e.g., the system reset signal, then hold or assert the other management signal in a specified state, e.g., a reset state.

As noted above with reference to FIG. 1B, in one embodiment, the microcontroller 102 may be situated between the system interface 106 (and therefore, the processor in the system) and the shared device 108, e.g., non-volatile memory, e.g., embedded flash memory, and thus may intercept communications between the system interface and the device 108.

The device 108 may include or be associated with an interface whereby access to the device is facilitated or implemented, and where this interface may be configured to direct or restrict access to the device to different system components, e.g., the processor or the microcontroller. In one embodiment, the microcontroller setting control signals to cause access to the shared device to be with the microcontroller comprises the microcontroller setting the interface to the shared device interface with the microcontroller. Said another way, the microcontroller may set one or more control signals to configure or redirect the shared device interface to grant exclusive access to the device to the microcontroller. In other embodiments, the switching of access to the shared device from the processor to the microcontroller may be performed via software, or a combination of software and hardware. Note that as used herein, the term "signal" may denote a hardware signal and/or a software signal, e.g., a software event, message, token, handshake, etc. Thus, the term "management signal" may refer to a hardware signal or a software signal.

Note that in a preferred embodiment, the microcontroller controls and manages power on reset (POR) for the system, and thus may respond to and/or generate signals or software events or messages related to power on reset or system reset (or other management signals) in a way that preempts access to the device by the main processor of the system, thereby facilitating sharing a device that otherwise could not be shared. In some embodiments, the change in system state may be invoked or indicated by a management signal, e.g., a system reset or POR signal, generated or received by the microcontroller. It should be noted, however, that in some embodiments, control of the direction of the interface, i.e., control of which component of the system has access to the device, may be independent of the management (e.g., system reset or POR) signal. In other words, the microcontroller may control whether the shared device interface is directed to the processor of the system, or to the microcontroller, independent of the management (e.g., system reset or POR) signal. Moreover, in some embodiments, instead of, or in addition to, hardware signals, the microcontroller may respond to software events, messages, tokens, handshakes, etc., that may

indicate when the system state is such that the microcontroller is assured safe access to the shared device, e.g., to non-volatile memory.

In 204, while the system is in the first state, i.e., the state where the microcontroller is insured safe access to the device, or can prohibit access by the processor in the system to the shared device, and has properly configured access to the device, the microcontroller may retrieve or exchange data with the device 108. For example, in embodiments where the system state is dependent on or indicated by a management signal, e.g., a system reset signal, the microcontroller may generate or receive the management signal, and hold the management signal in this state, i.e., may maintain assertion of the management signal, and, while this signal is asserted, may read, write, acquire, or otherwise exchange (e.g., via invocation of a read or write by some other component of the system), data or program instructions from, to, or with, the device. In one embodiment, while the system is held in the first state, the microcontroller may fetch program instructions/data from the non-volatile memory and load the program instructions/data into a memory of the microcontroller.

In 206, once the controller has completed the data retrieval or exchange with the device or has determined that it may no longer be assured of safe access, the microcontroller may reconfigure the interface through control signals to enable access by the processor in the system, e.g., via a switch that controls access to the device, as will be described below in more detail.

In 208, once the microcontroller has configured the device to enable access by the processor in the system, the microcontroller may change or allow change of the system state, thereby permitting the processor in the system to access the shared device. For example, in embodiments where the system state is dependent on or indicated by a management signal, e.g., a system reset signal, the microcontroller may de-assert the management signal, thereby putting the system into a state that allows access to the shared device by the processor of the system. Said another way, after the program instructions/data have been loaded, the microcontroller may switch access to the shared device, e.g., the non-volatile memory, from the microcontroller to the processor, and release the system from the first state.

Thus, using embodiments of the above-described method, the microcontroller 102 may preempt or suspend access to the device by the processor, e.g., the main CPU, of the system, or otherwise configure safe access to the device by the microcontroller; retrieve or otherwise exchange data or program instructions with the device; then return access to the device to the processor, or otherwise relinquish exclusive or safe access to the device by the microcontroller.

It should be noted that in various other embodiments where the microcontroller manages or controls operating states of the system (e.g., reset, POR, sleep, and other states related to specific system activities where it is known by the microcontroller that the system cannot or will not access the device), the microcontroller may monitor (or control) such system states, and take advantage of these states to access the device safely, e.g., to take control and use the shared device. Note that these system states should be such that the microcontroller either manages when the system will exit these states or is provided sufficient warning as to when the system will again need control of the shared interface (to the device).

As one example, when a system is being shut down, the microcontroller may take over the device interface, i.e., may redirect the device interface to allow (only) access by the microcontroller, before power-off, i.e., before turning off the final power rail. In this way, writes or other exchanges with

11

the shared device may be “saved up” or queued, and their completion (e.g., by the microcontroller) may be guaranteed before final power off.

Thus, the microcontroller **102** may gracefully share access to the device **108** with the processor of the system. It should be noted that in other embodiments, additional microcontrollers may also share the device with the host or main system CPU, using the techniques described above. In further embodiments, such sharing of devices may also be used in other ways. For example, in some embodiments, a plurality of processors (e.g., microcontrollers) may share such a device, where each processor may in turn suspend or prevent access to the device by the other processors while it accesses the device.

#### Other Embodiments

Below are described more specific embodiments of the above-described general method of FIG. 2 that are particularly directed to the sharing of non-volatile memory between a main processor and a microcontroller in a system. In other words, in the below embodiments, the shared device comprises a non-volatile memory device. It should be noted, however, that the embodiments of FIG. 3 are meant to be exemplary only, and are not intended to limit the type or use of the shared device to any particular form or function. FIG. 3—Method for Sharing Non-Volatile Memory Between Processors in a System

FIG. 3 is a high-level flowchart of a method for sharing non-volatile memory between processors, e.g., between a processor and a microcontroller in a system, e.g., a computer system, according to one embodiment. It should be noted, however, that in other embodiments, any other types of memory may be used as desired, e.g., RAM, ROM, etc. In various embodiments, some of the method elements shown may be performed concurrently, in a different order than shown, or may be omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

In **302**, in response to a system state change, such as, for example, power on reset (POR), system reset, or sleep mode, where a microcontroller, e.g., microcontroller **102**, in a system is assured of safe access to a non-volatile memory, or can prevent the system from entering a state where a processor in the system can access the non-volatile memory, the microcontroller may set control signals or utilize software to cause access to the non-volatile memory to be with the microcontroller. In other words, when the system is changed to a state, e.g., a first state, in which the microcontroller **102** is guaranteed the ability to access the non-volatile memory safely, e.g., without interruption or interference by the processor, the microcontroller may hold the system in the state and switch access to the non-volatile memory from the processor to the microcontroller. Examples of this first state may include one or more of: a power on reset (POR) state, a system reset state, or a sleep state, among others.

Holding the system in this first state may prohibit the processor in the system from accessing the non-volatile memory in the system. Similarly, the state change may be invoked or caused by a management signal, such as a POR, system reset, sleep, signal, etc., which may include a software signal, event, token, handshake, etc. In one embodiment, holding the system in the first state may include holding a management signal in a corresponding state, e.g., holding a system reset signal in a reset state.

Note that the management signal, e.g., the POR and/or the system reset, and/or any other type of management signal

12

used, may or may not be invoked by the microcontroller. Thus, for example, in response to a received management signal, e.g., POR signal, or in response to generating the management signal, e.g., the POR signal, the microcontroller may generate or receive another management signal, e.g., the system reset signal, then hold or assert the other management signal in a specified state, e.g., a reset state.

As noted above with reference to FIG. 1B, in one embodiment, the microcontroller **102** may be situated between the system interface **106** (and therefore, the processor in the system) and the shared device **108**, e.g., non-volatile memory, e.g., embedded flash memory, and thus may intercept communications between the system interface and the device, e.g., the non-volatile memory. As will be described below in more detail, in some embodiments, when the system is held in the first state, e.g., when the system reset signal is held in the reset state, the interface to the non-volatile memory may be configured by the microcontroller to preclude the processor in the system from accessing the non-volatile memory, e.g., via a switch.

FIG. 4 illustrates an exemplary embodiment of the microcontroller **102** of FIG. 1B, where the device **108** comprises non-volatile memory. As FIG. 4 shows, in this embodiment, the microcontroller **102** includes a system reset line **403** for communicating a reset signal, i.e., a system reset signal (or other management signal), e.g., to or from the system interface **106** over the system management bus **101** (see FIG. 1B). While many of the embodiments described herein reference a reset signal, it should be noted that in other embodiments, as mentioned above, other signals may be used as desired, e.g., any signal that corresponds with or invokes a suspension of non-volatile memory access by the CPU (or other specified processor) of the system, or which indicates a system state in which the microcontroller may have safe access to the non-volatile memory, including software events, tokens, handshakes, signals, etc. Note that FIG. 4 only illustrates features of the microcontroller that relate directly to embodiments of the present invention, i.e., components not necessary to understand the present invention have been omitted for brevity and simplicity.

As discussed above, in various embodiments, the switching of access to the shared device from the processor to the microcontroller may be performed via hardware, via software, or via a combination of software and hardware. As also discussed above, the term “signal” may denote a hardware signal and/or a software signal, e.g., a software event, message, token, handshake, etc., i.e., the term “management signal” may refer to a hardware signal or a software signal.

As also indicated above, in a preferred embodiment, the microcontroller controls and manages power on reset (POR) for the system, and thus may respond to and/or generate signals or software events or messages related to power on reset or system reset (or other management signals) in a way that preempts access to the device by the main processor of the system, thereby facilitating sharing a non-volatile memory that otherwise could not be shared. In some embodiments, the change in system state may be invoked or indicated by a management signal, e.g., a system reset or POR signal, generated or received by the microcontroller. It should be noted, however, that in some embodiments, control of the direction of the interface, i.e., control of which component of the system has access to the non-volatile memory, may be independent of the management (e.g., system reset or POR) signal. In other words, the microcontroller may control whether the non-volatile memory interface is directed to the processor of the system, or to the microcontroller, independent of the management (e.g., system reset or POR) signal.



13

Moreover, as noted above, in some embodiments, instead of, or in addition to, hardware signals, the microcontroller may respond to software events, messages, tokens, handshakes, etc., that may indicate when the system state is such that the microcontroller is assured safe access to the non-volatile memory.

In **304**, while the system is in the first state, i.e., the state where the microcontroller is insured safe access to the device, or can prohibit access by the processor in the system to the non-volatile memory, and has properly configured access to the non-volatile memory, the microcontroller may retrieve or exchange data with the non-volatile memory. For example, while the system is held in the first state, the microcontroller may fetch program instructions/data from the non-volatile memory and load the program instructions/data into a memory of the microcontroller.

In one embodiment, holding the system in the first state may include holding a system reset signal (or other management signal) in the reset state, e.g., asserting system reset. While the system reset signal is held in the reset state, and the interface to the non-volatile memory is configured to only allow access by the microcontroller, the microcontroller may fetch program instructions/data from the non-volatile memory and load the program instructions/data into a memory of the microcontroller, or otherwise access the non-volatile memory. In other words, the microcontroller may hold the system reset signal in the reset state (e.g., on the system management bus **101** via system reset line **403**), i.e., may maintain assertion of the system reset signal, and, while the system reset signal is asserted, may fetch, i.e., read, acquire, or otherwise receive (e.g., via invocation of a read by some other component of the system), program instructions/data from the non-volatile memory and load or write the program instructions/data into the microcontroller's memory. Note that the microcontroller's memory is preferably RAM (random access memory), although any type of re-writable memory may be used as desired. As noted above, in some embodiments, the program instructions/data may include program firmware code, that may, for example, be usable by the embedded microcontroller for startup or reset functionality, e.g., for initializing the microcontroller, among other uses. In some embodiments, the microcontroller may be or include a peripheral device controller for a computer system. For example, in one exemplary embodiment, the embedded microcontroller may be a keyboard controller, and the non-volatile memory device **108** may store keyboard controller (KBC) code, although in other embodiments, any other program instructions/code or data may be used as desired.

Referring again to FIG. 4, as may be seen, in this embodiment, the microcontroller **102** may include or couple to the host memory bus **105**, which, as indicated in FIG. 1B, may couple to the system interface **106**, and may also include or couple to device bus **107**, in this embodiment, a non-volatile memory bus **107**, which may couple to the shared non-volatile memory device **108**. Note that in the embodiment shown, the host memory bus **105** and the non-volatile memory bus **107** each includes three lines, one each for data, clock, and CSb signals, where CSb stands for Chip Select bar, which is a low true version of the Chip Select, and as such is part of the address structure of the block. In one embodiment, the host memory bus **105** and the non-volatile memory bus **107** are SPI (serial peripheral interface) buses, although other protocols may be used as desired. Note that in some embodiments, the host memory bus **105** and the non-volatile memory bus **107** may be considered the same bus, e.g., a memory bus, where the microcontroller **102** is inserted to switchably intercept and possibly redirect signals on the bus(es).

14

As FIG. 4 indicates, in this embodiment, the microcontroller **102** includes a switch **404** controlled by switch control **406**, for providing switching functionality for the host memory bus **105** and the non-volatile memory bus **107**, e.g., for the memory bus. In some embodiments, the switch **404** may comprise a Q switch, although it should be noted that the term "Q switch" may refer not only to actual Q switches, but to functional equivalents or analogs, as well. Said another way, the switch **404** may be a "conceptual Q switch", that may perform the functionality of a Q switch (or functionality similar to a Q switch), but which may implement this functionality in ways different from a Q switch. Note that in the embodiment of FIG. 4, the switch **404** includes a respective switch for each of the memory bus lines, as indicated by the "X3" (times three) label. Thus, each line on the memory bus may be switched independently, although in preferred embodiments, these lines may all be switched together, thereby suspending or inhibiting communications between the host interface **106** and the non-volatile memory **108**. Of course, in other embodiments, any numbers of bus lines/switches may be used as desired.

It should be noted that in other embodiments, the microcontroller **102** may not include the switch **404**, but rather may couple to a switch that is external to the microcontroller. One benefit of this approach is that it may save pins on the microcontroller.

As FIG. 4 further indicates, each line on the memory bus may be coupled to respective buffers for communicating data/signals between the memory bus (i.e., the device bus) and the microcontroller **102**, where, as noted above, most of the internal components of the microcontroller **102** have been omitted from the figure. Thus, as FIG. 4 shows, the microcontroller **102** may be operable to switchably preempt and inhibit signal flow on the memory bus (i.e., the host memory bus **105** and the non-volatile memory bus **107**) between the host interface **106** and the non-volatile memory **108**, e.g., by placing the switches into a high-impedance state, and to fetch instructions/data from the non-volatile memory **108** while communication between the host interface **106** and the non-volatile memory **108** is suspended. Thus, the microcontroller may operate to configure the interface to the device (e.g., the switch) to regulate and enforce safe access to the device.

In **306**, once the controller has completed the data retrieval or exchange with the non-volatile memory, or has determined that it may no longer be assured of safe access, the microcontroller may reconfigure the interface through control signals to enable access by the processor in the system, e.g., via a switch that controls access to the non-volatile memory.

In a more specific embodiment, after the microcontroller has loaded the program instructions/data from the non-volatile memory, the microcontroller may release the system reset signal from the reset state, thereby permitting the processor in the system access to the non-volatile memory. In other words, upon completion of the fetch of instructions/data from the non-volatile memory **108** by the microcontroller **102**, the microcontroller may terminate the suspension or preemption of communications between host interface **106** and the non-volatile memory **108**, thereby allowing resumption of such communications, e.g., via switch **404**.

Thus, using embodiments of the above-described method, the microcontroller **102** may preempt or suspend communications with the non-volatile memory by the main CPU and read instructions/data from the non-volatile memory, e.g., to initialize the microcontroller **102** and/or perform other functionality prior to, or independent from, operation of the main system, i.e., the main CPU, thus sharing the non-volatile memory with the main system. For example, in some embodi-

15

ments, the microcontroller may, in addition to initialization code/data, may also fetch program instructions and/or data related to pre-boot security operations, and may operate to perform such pre-boot security operations prior to boot-up of the main system, thereby providing additional security to the system.

FIG. 5 is a signal timing diagram that illustrates timing aspects of the above method, according to one embodiment. As FIG. 5 shows, in this particular embodiment, a VCC, i.e., power supply voltage, is provided, followed by a VCC PWRGD ("power good") signal indicating that the power supply voltage is valid, i.e., is at a valid level and is stable. It should be noted that the particular names or labels for these signals or voltages are exemplary only, and are not intended to limit the signals or voltages used to any particular names or representations.

As indicated, once the supply voltage is stable, the microcontroller 102 may hold the (generated or received) system reset signal in a reset state, here shown in the "low" state (e.g., a "low" or negative assertion state), during which time the microcontroller 102 may fetch and load instructions/data, e.g., controller firmware code, from the non-volatile memory, and during which, due to the system reset signal being held in the reset state, the host, i.e., the main system CPU, is inhibited from accessing the non-volatile memory. As FIG. 5 also shows, once this loading is done, the reset signal is de-asserted (shown in FIG. 5 as being put into the "high" state), after which the non-volatile memory is accessible by the host or main system CPU.

Thus, the microcontroller 102 may gracefully share use of the main system's non-volatile memory. It should be noted that in other embodiments, additional microcontrollers may also share this non-volatile memory (or device) with the host or main system CPU, using the techniques described above. In further embodiments, such sharing of non-volatile memory may also be used in other ways. For example, in some embodiments, a plurality of microcontrollers may share a non-volatile memory (e.g., that is not used by the main system CPU), where each microcontroller may in turn suspend or prevent access to the non-volatile memory by the other microcontrollers while it loads instructions/data from the memory. Thus, various embodiments described with reference to FIG. 3 may allow an embedded microcontroller to upload its firmware and data from a non-volatile memory device that is already present in the system and therefore save the cost and avoid issues associated with a private flash or ROM storage for the microcontroller.

Note that in various embodiments, the system changing to a state where the microcontroller is assured safe access to the non-volatile memory (or device) may be performed or mediated by software or hardware. In other words, in some embodiments, the state change may be in response to hardware signals, such as POR signals, etc., but in other embodiments, the state change may be in response to software events, tokens, software conditions, software messages, software handshakes, and so forth. Said another way, embodiments of the present invention are intended to cover virtually any case where the embedded controller can determine that it can safely take control of the shared non-volatile memory, whether the "safe" system state is mediated or signaled via software or hardware.

Thus, various embodiments of the invention described herein may allow an embedded microcontroller and a processor in a system to share a device, e.g., a non-volatile memory, that is not designed to be shared.

16

We claim:

1. A system, comprising:

a non-volatile memory;

a processor; and

a microcontroller coupled to the non-volatile memory and to the processor, wherein the microcontroller is configured to:

in response to a change in system state to a first state wherein the microcontroller is assured safe access to the non-volatile memory, hold the system in the first state and switch access to the non-volatile memory from the processor to the microcontroller;

while the system is held in the first state, fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller; and

after the program instructions or data have been loaded, switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state.

2. The system of claim 1, wherein the change in system state to the first state is performed by software.

3. The system of claim 1, wherein the first state comprises one or more of:

a power on reset (POR) state;

a system reset state; or

a sleep state.

4. The system of claim 1, wherein the program instructions or data comprise firmware code or data for the microcontroller.

5. The system of claim 1, the microcontroller is a peripheral device controller for a computer system.

6. A method for sharing a non-volatile memory between a processor and a microcontroller in a system, comprising:

in response to a change in system state to a first state wherein the microcontroller is assured safe access to the non-volatile memory, holding the system in the first state and switching access to the non-volatile memory from the processor to the microcontroller;

while the system is held in the first state, fetching program instructions or data from the non-volatile memory and loading the program instructions or data into a memory of the microcontroller; and

after the program instructions or data have been loaded, switching access to the non-volatile memory from the microcontroller to the processor, and releasing the system from the first state.

7. The method of claim 6, wherein the change in system state to the first state is performed by software.

8. The method of claim 6, wherein the first state comprises one or more of:

a power on reset (POR) state;

a system reset state; or

a sleep state.

9. The method of claim 6, wherein the program instructions or data comprise firmware code or data for the microcontroller.

10. The method of claim 6, the microcontroller is a peripheral device controller for a computer system.

11. A system, comprising:

a non-volatile memory;

a processor; and

a microcontroller coupled to the non-volatile memory and to the processor, wherein the microcontroller is configured to:

in response to a power on reset, hold a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory;

17

while the system reset signal is held in the reset state, fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller; and  
 after the program instructions or data have been loaded, 5 permit the processor in the system access to the non-volatile memory, and release the system reset signal from the reset state.

12. The system of claim 11, wherein the program instructions or data comprise firmware code or data for the microcontroller. 10

13. The system of claim 12, the microcontroller is a peripheral device controller for a computer system.

14. The system of claim 11, wherein the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to 15 fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus.

15. The system of claim 14, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable to switchably intercept or pass 20 communications between the processor and the non-volatile memory.

16. The system of claim 15, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor 25 and the non-volatile memory.

17. A method for sharing a non-volatile memory between a processor and a microcontroller in a system, comprising: 30  
 in response to a power on reset, holding a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory;

18

while the system reset signal is held in the reset state, fetching program instructions or data from the non-volatile memory and loading the program instructions or data into a memory of the microcontroller; and  
 after the program instructions or data have been loaded, 5 permitting the processor in the system access to the non-volatile memory, and releasing the system reset signal from the reset state.

18. The method of claim 17, wherein the program instructions or data comprise firmware code or data for the microcontroller. 10

19. The method of claim 17, the microcontroller is a peripheral device controller for a computer system.

20. The method of claim 17, wherein the microcontroller is coupled to the processor via a system reset bus, and wherein the microcontroller is operable to assert the system reset signal over the system reset bus.

21. The method of claim 17, wherein the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to 15 fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus.

22. The method of claim 21, wherein the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable switchably intercept or pass 20 communications between the processor and the non-volatile memory.

23. The method of claim 22, wherein the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor 25 and the non-volatile memory.

\* \* \* \* \*

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Ex Parte Reexamination on U.S. Patent No. **7,930,576**, issued on April 19, 2011

|             |                                |   |                        |
|-------------|--------------------------------|---|------------------------|
| Inventors : | Harris; Ian F                  | ) |                        |
|             | Dutton; Drew J.                | ) |                        |
|             |                                | ) |                        |
| Appl. No. : | <b>11/958,601</b>              | ) |                        |
|             |                                | ) | Ex Parte Reexamination |
| Filed :     | December 18, 2007              | ) |                        |
|             |                                | ) |                        |
| For :       | SHARING NON-SHARABLE DEVICES   | ) |                        |
|             | BETWEEN AN EMBEDDED CONTROLLER | ) |                        |
|             | AND A PROCESSOR IN A COMPUTER  | ) |                        |
|             | SYSTEM                         | ) |                        |

---

**REQUEST FOR EX PARTE REEXAMINATION UNDER 37 C.F.R. § 1.510**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Dear Sir:

Reexamination under 35 U.S.C. § 302-307 and 37 C.F.R. § 1.510 is hereby requested on United States Patent No. 7,930,576 naming Harris; Ian F. and Dutton; Drew J. as inventors (hereinafter "the '576 patent"). The '576 patent issued on April 19, 2011. The named assignee of the patent is **Standard Microsystems Corporation**. The '576 patent has not expired and remains in effect.

**CLAIMS FOR WHICH REEXAMINATION IS REQUESTED**

Reexamination is requested on **claims 1-23** of the '576 patent in light of United State Patent No. US 6,161,162 (DeRoo et al., hereinafter DeRoo). DeRoo raises a substantial new question of patentability that should be addressed during reexamination.

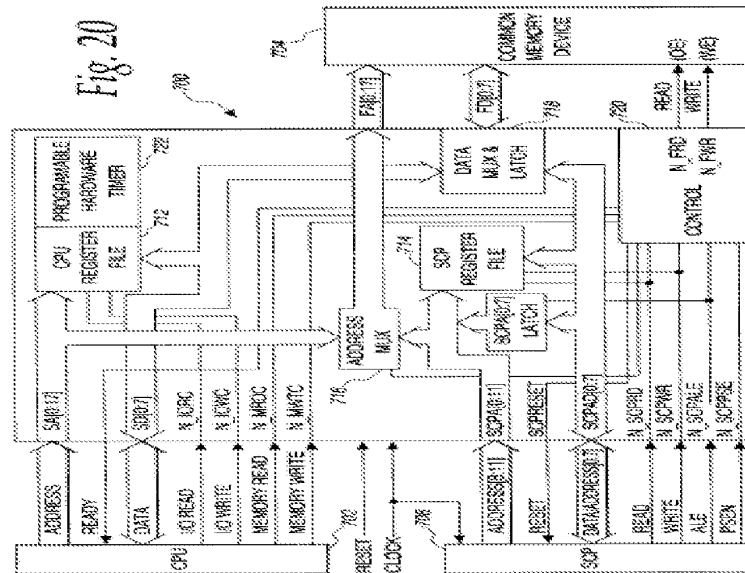
**SUBSTANTIAL NEW QUESTION OF PATENTABILITY**

Fig. 23

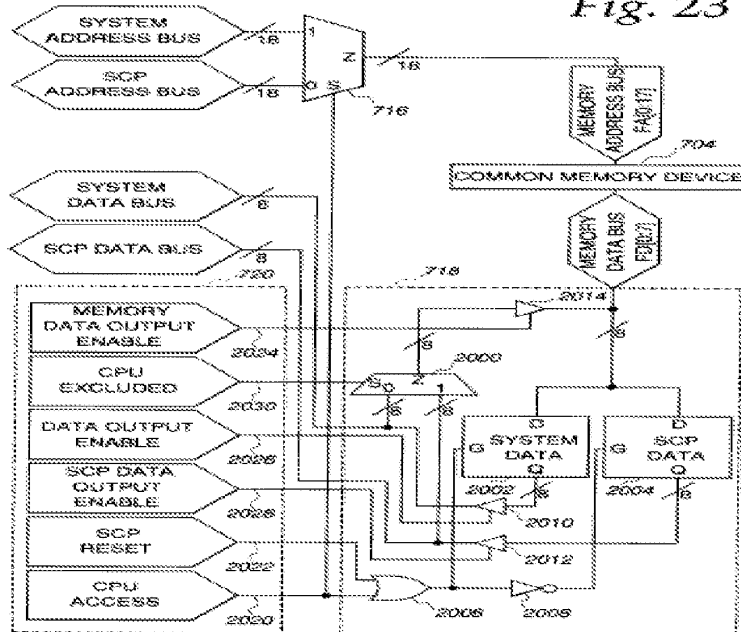
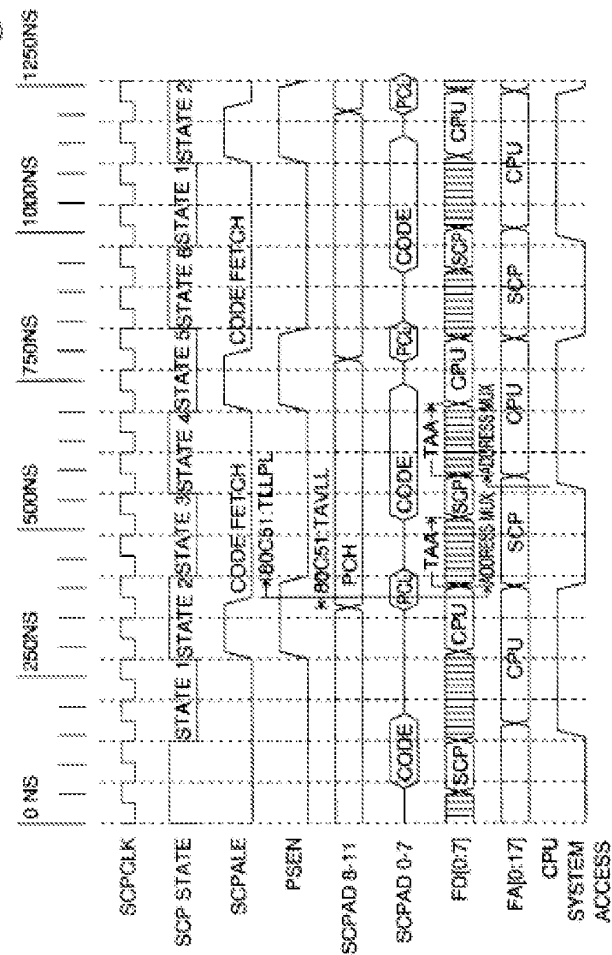


Fig. 24



Figs. of DeRoo



**Part I:**

Requester respectfully submits that claims 1-2, 4-7 and 9-10 of the '576 patent are unpatentable over DeRoo under 35 U.S.C. 102(b), claims 3, 8 and 11-23 of the '576 patent are unpatentable over DeRoo under 35 U.S.C. 103(a). DeRoo is a continuation of 08/217,958 which was filed on March 25 1994. Since DeRoo was published more than one year prior to the application for the '576 patent was filed on December 18, 2007.

With regard to claim 1 of US 7,930,576, DeRoo discloses a system (*title, "Multiprocessor system for enabling shared access to a memory"; and FIG. 20*), comprising:

a non-volatile memory (*FIG 20, common memory device 704*);

a processor (*FIG20, CPU 702*) ; and

a microcontroller (*FIG. 20, SCP 706 and HUI 700*) coupled to the non-volatile memory and to the processor, wherein the microcontroller is configured to: in response to a change in system state to a first state (*FIG. 24, when the CPU SYSTEM ACCESS signal is in low level*) wherein the microcontroller is assured safe access to the non-volatile memory, hold the system in the first state and switch access to the non-volatile memory from the processor to the microcontroller; while the system is held in the first state, fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller; and after the program instructions or data have been loaded, switch access to the non-volatile memory from the microcontroller to the processor, and release the system from the first state. (*FIG.20; "The HUI solves this problem by providing an interface to enable both the SCP and the CPU to share a common memory device", col. 36 lines 30-32; "Reads and writes by*

*the CPU 702 to the HUI 700 are under the control of various CPU control signals including an I/O read strobe N\_IORC, an I/O write strobe N\_IOWC, a memory data read strobe N\_MRDC and a memory data write strobe N\_MWTC”, col. 36, lines 56-60. “Communication between the SCP 706 and the CPU 702, and between the SCP 706 and the common memory device 704 is under the control of various control signals including a memory read strobe N\_SCPRD, a memory write strobe N\_SCPWR, and address latch enable N\_SCPALE and a program store enable N\_CPPSE”, col. 36-37, lines 66-4. Moreover, DeRoo also discloses a CONTROL 720 in FIG. 20 for accessing the common memory device 704 from either the CPU register file 712 or the SCP register file 714, col. 37, lines 6-23 ).*

*In the other hand, DeRoo discloses “[t]he HUI 700 allows both the CPU 702 and the SCP 706 to share a common memory device 704” (col. 81, lines 40-42) and “the BIOS, as well as the SCP code, can be stored in a single memory device” (col. 81, lines 42-43). DeRoo also discloses “[f]ull access is provided to both the CPU 702, as well as the SCP 706, utilizing a time based interleaving method as illustrated in FIG. 24” (col. 81, lines 50-52; FIG. 24) and “HUI 700 multiplexes access to the common device between the CPU 702 and the SCP 706” (col. 82, lines 8-9; FIG. 23), that is, DeRoo discloses **“switch access to the non-volatile memory from the microcontroller to the processor”** of claim 1.*

With regard to claim 2 of US 7,930,576, DeRoo discloses the change in system state to the first state is performed by software. (*“the HUI 700 allows the SCP 706 to communicate with the CPU 702 and the common memory device 704 by way of an SCP register file 714” col. 36, lines 42-44. It is well known that to write code into a register by a software program, and the SCP*

*register file 714 can be controlled by a software program for establish between the SCP 706 and the common memory device 704 is well known for a person skilled in the art)*

With regard to claim 4 of US 7,930,576, DeRoo discloses the program instructions or data comprise firmware code or data for the microcontroller. (*“Write external RAM (program &or data)” and “Read from External Program/Data memory”, col. 22, Table VII, last 2 lines. Additionally, it is well known to store firmware code or data into a memory in a computer system).*

With regard to claim 5 of US 7,930,576, DeRoo discloses the microcontroller is a peripheral device controller for a computer system (*“The SCP 20 monitors the keyboard 22”, col. 4, line 22).*

With regard to claim 6 of US 7,930,576, DeRoo discloses a method for sharing a non-volatile memory (*FIG 20, common memory device 704*) between a processor (*FIG20, CPU 702*) and a microcontroller (*FIG. 20, SCP 706 and HUI 700*) in a system (*title, “Multiprocessor system for enabling shared access to a memory”; and FIG. 20*), comprising: in response to a change in system state to a first state (*FIG. 24, when the CPU SYSTEM ACCESS signal is in low level*) wherein the microcontroller is assured safe access to the non-volatile memory, holding the system in the first state and switching access to the non-volatile memory from the processor to the microcontroller; while the system is held in the first state, fetching program instructions or data from the non-volatile memory and loading the program instructions or data into a memory of the

microcontroller; and after the program instructions or data have been loaded, switching access to the non-volatile memory from the microcontroller to the processor, and releasing the system from the first state (FIG.20; “The HUI solves this problem by providing an interface to enable both the SCP and the CPU to share a common memory device”, col. 36 lines 30-32; “ Reads and writes by the CPU 702 to the HUI 700 are under the control of various CPU control signals including an I/O read strobe N\_IORC, an I/O write strobe N\_IOWC, a memory data read strobe N\_MRDC and a memory data write strobe N\_MWTC”, col. 36, lines 56-60. “Communication between the SCP 706 and the CPU 702, and between the SCP 706 and the common memory device 704 is under the control of various control signals including a memory read strobe N\_SCPRD, a memory write strobe N\_SCPWR, and address latch enable N\_SCPALE and a program store enable N\_CPPSE”, col. 36-37, lines 66-4. Moreover, DeRoo also discloses a CONTROL 720 in FIG. 20 for accessing the common memory device 704 form either the CPU register file 712 or the SCP register file 714, col. 37, lines 6-23 ).

In the other hand, DeRoo discloses “[t]he HUI 700 allows both the CPU 702 and the SCP 706 to share a common memory device 704” (col. 81, lines 40-42) and “the BIOS, as well as the SCP code, can be stored in a single memory device” (col. 81, lines 42-43). DeRoo also discloses “[f]ull access is provided to both the CPU 702, as well as the SCP 706, utilizing a time based interleaving method as illustrated in FIG. 24” (col. 81, lines 50-52; FIG. 24) and “HUI 700 multiplexes access to the common device between the CPU 702 and the SCP 706” (col. 82, lines 8-9; FIG. 23), that is, DeRoo discloses “**switch access to the non-volatile memory from the microcontroller to the processor**” of claim 6.

With regard to claim 7 of US 7,930,576, DeRoo discloses the change in system state to the first state is performed by software. (*“the HUI 700 allows the SCP 706 to communicate with the CPU 702 and the common memory device 704 by way of an SCP register file 714” col. 36, lines 42-44. It is well known that to write code into a register by a software program, and the SCP register file 714 can be controlled by a software program for establish between the SCP 706 and the common memory device 704 is well known for a person skilled in the art*)

With regard to claim 9 of US 7,930,576, DeRoo discloses the program instructions or data comprise firmware code or data for the microcontroller. (*“Write external RAM (program &or data)” and “Read from External Program/Data memory”, col. 22, Table VII, last 2 lines. Additionally, it is well known to store firmware code or data into a memory in a computer system*).

With regard to claim 10 of US 7,930,576, DeRoo discloses the microcontroller is a peripheral device controller for a computer system (*“The SCP 20 monitors the keyboard 22”, col. 4, line 22*).

With regard to claim 3 of US 7,930,576, DeRoo discloses the first state comprises one or more of: a power on reset (POR) state; a system reset state; or a sleep state. *Please notice here, DeRoo fails to disclose the first state comprises one or more of power on reset state, a system reset state or a sleep state explicitly. But it is well known to set the first state when the processor is in one or more of power on reset state, a system reset state or a sleep state.*

With regard to claim 8 of US 7,930,576, DeRoo discloses the first state comprises one or more of: a power on reset (POR) state; a system reset state; or a sleep state. *Please notice here, DeRoo fails to disclose the first state comprises one or more of power on reset state, a system reset state or a sleep state explicitly. But it is well known to set the first state when the processor is in one or more of power on reset state, a system reset state or a sleep state.*

As detailed above, claims 1-2, 4-7 and 9-10 of the '576 patent are unpatentable over DeRoo under 35 U.S.C. 102(b), claims 3 and 8 of the '576 patent are unpatentable over DeRoo under 35 U.S.C. 103(a).

With regard to claim 11 of US 7,930,576, DeRoo discloses a system, comprising:

a non-volatile memory (*FIG 20, common memory device 704*);

a processor(*FIG20, CPU 702*); and

a microcontroller (*FIG. 20, SCP 706 and HUI 700*) coupled to the non-volatile memory and to the processor, wherein the microcontroller is configured to: in response to a power on reset (*DeRoo fails to disclose the first state is when the system is in power on reset state, but it is well known to set the first state when the system is in the power on reset state*) , hold a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory (*FIG 20, the CONTROL 720 can only access the common memory device 704 for one of SCP 706 and CPU 702. When SCP 706 accesses the common memory device 704 through the CONTROL 720, the CPU 702 is prohibited to access the common memory device 704*);

while the system reset signal is held in the reset state, fetch program instructions or data from the non-volatile memory and load the program instructions or data into a memory of the microcontroller; and after the program instructions or data have been loaded, permit the processor in the system access to the non-volatile memory, and release the system reset signal from the reset state. (FIG.20; *“The HUI solves this problem by providing an interface to enable both the SCP and the CPU to share a common memory device”, col. 36 lines 30-32; “ Reads and writes by the CPU 702 to the HUI 700 are under the control of various CPU control signals including an I/O read strobe N\_IORC, an I/O write strobe N\_IOWC, a memory data read strobe N\_MRDC and a memory data write strobe N\_MWTC”, col. 36, lines 56-60. “Communication between the SCP 706 and the CPU 702, and between the SCP 706 and the common memory device 704 is under the control of various control signals including a memory read strobe N\_SCPRD, a memory write strobe N\_SCPWR, and address latch enable N\_SCPALE and a program store enable N\_CPPSE”, col. 36-37, lines 66-4. Moreover, DeRoo also discloses a CONTROL 720 in FIG. 20 for accessing the common memory device 704 form either the CPU register file 712 or the SCP register file 714, col. 37, lines 6-23 ).*

*In the other hand, DeRoo discloses “[t]he HUI 700 allows both the CPU 702 and the SCP 706 to share a common memory device 704” (col. 81, lines 40-42) and “the BIOS, as well as the SCP code, can be stored in a single memory device” (col. 81, lines 42-43). DeRoo also discloses “[f]ull access is provided to both the CPU 702, as well as the SCP 706, utilizing a time based interleaving method as illustrated in FIG. 24” (col. 81, lines 50-52; FIG. 24) and “HUI 700 multiplexes access to the common device between the CPU 702 and the SCP 706” (col. 82, lines*

8-9; FIG. 23). That is, DeRoo discloses to switch the accessing to the memory from the microcontroller to the processor in the claim 11.

With regard to claim 12 of US 7,930,576, DeRoo discloses the program instructions or data comprise firmware code or data for the microcontroller (*“Write external RAM (program &or data)” and “Read from External Program/Data memory”, col. 22, Table VII, last 2 lines. Additionally, it is well known to store firmware code or data into a memory in a computer system).*

With regard to claim 13 of US 7,930,576, DeRoo discloses the microcontroller is a peripheral device controller for a computer system (*“The SCP 20 monitors the keyboard 22”, col. 4, line 22).*

With regard to claim 14 of US 7,930,576, DeRoo discloses the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus (*FIG. 20, FD[0:7]*).

With regard to claim 15 of US 7,930,576, DeRoo discloses the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable to switchably intercept or pass communications between the processor and the non-volatile memory (*FIG. 20, address mux 716, data mux & latch 718).*



With regard to claim 16 of US 7,930,576, DeRoo discloses the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor and the non-volatile memory (*FIGs. 20 and 23, address mux 716, data mux & latch 718. “the SCP 706 can gain exclusive access to the common memory device 704 by asserting a CPU EXCLUDED signal (i.e., setting bit 1 of the flash interface control register FLASH\_CTRL). During such a condition, the data multiplexer 718 blocks the system data bus SD[0:7] from the path of the memory data bus FD[0:7]”, col. 82, lines 32-39. The address mux 716 and the data mux and latch 718 are controlled by the outputs of control logic 720, and the values of the registers in the control logic 720 can be set by either SCP or CPU).*

With regard to claim 17 of US 7,930,576, DeRoo discloses a method for sharing a non-volatile memory between a processor and a microcontroller in a system, comprising: in response to a power on reset, holding a system reset signal in a reset state, thereby prohibiting the processor from accessing the non-volatile memory; while the system reset signal is held in the reset state, fetching program instructions or data from the non-volatile memory and loading the program instructions or data into a memory of the microcontroller; and after the program instructions or data have been loaded, permitting the processor in the system access to the non-volatile memory, and releasing the system reset signal from the reset state (*FIGs. 20 and 23, address mux 716, data mux & latch 718. “the SCP 706 can gain exclusive access to the common memory device 704 by asserting a CPU EXCLUDED signal (i.e., setting bit 1 of the flash interface control register*

*FLASH\_CTRL*). During such a condition, the data multiplexer 718 blocks the system data bus *SD[0:7]* from the path of the memory data bus *FD[0:7]*”, col. 82, lines 32-39. The address mux 716 and the data mux and latch 718 are controlled by the outputs of control logic 720, and the values of the registers in the control logic 720 can be set by either SCP or CPU).

In the other hand, DeRoo discloses “[t]he HUI 700 allows both the CPU 702 and the SCP 706 to share a common memory device 704” (col. 81, lines 40-42) and “the BIOS, as well as the SCP code, can be stored in a single memory device” (col. 81, lines 42-43). DeRoo also discloses “[f]ull access is provided to both the CPU 702, as well as the SCP 706, utilizing a time based interleaving method as illustrated in FIG. 24” (col. 81, lines 50-52; FIG. 24) and “HUI 700 multiplexes access to the common device between the CPU 702 and the SCP 706” (col. 82, lines 8-9; FIG. 23). That is, DeRoo discloses to switch the accessing to the memory from the microcontroller to the processor in the claim 17.

With regard to claim 18 of US 7,930,576, DeRoo discloses the program instructions or data comprise firmware code or data for the microcontroller (“Write external RAM (program &or data)” and “Read from External Program/Data memory”, col. 22, Table VII, last 2 lines. Additionally, it is well known to store firmware code or data into a memory in a computer system).

With regard to claim 19 of US 7,930,576, DeRoo discloses the microcontroller is a peripheral device controller for a computer system (“The SCP 20 monitors the keyboard 22”, col. 4, line 22).

With regard to claim 20 of US 7,930,576, DeRoo discloses the microcontroller is coupled to the processor via a system reset bus, and wherein the microcontroller is operable to assert the system reset signal over the system reset bus (*FIG. 20, RESET line for SCP*).

With regard to claim 21 of US 7,930,576, DeRoo discloses the microcontroller is coupled to the non-volatile memory via a non-volatile memory bus, and wherein the microcontroller is operable to fetch the program instructions or data from the non-volatile memory via the non-volatile memory bus (*FIGs. 20 and 23, FD[0:7]*).

With regard to claim 22 of US 7,930,576, DeRoo discloses the microcontroller is coupled to the processor via a host memory bus, and wherein the microcontroller is operable switchably intercept or pass communications between the processor and the non-volatile memory (*FIGs. 20 and 23*) .

With regard to claim 23 of US 7,930,576, DeRoo discloses the microcontroller comprises or is coupled to a switch, interposed between the non-volatile memory bus and the host memory bus, wherein the switch is controllable by the microcontroller to switchably intercept or pass communications between the processor and the non-volatile memory (*FIGs. 20 and 23, address mux 716, data mux & latch 718. “the SCP 706 can gain exclusive access to the common memory device 704 by asserting a CPU EXCLUDED signal (i.e., setting bit 1 of the flash interface control register FLASH\_CTRL). During such a condition, the data multiplexer 718 blocks the system data*

*bus SD[0:7] from the path of the memory data bus FD[0:7]”, col. 82, lines 32-39. The address mux 716 and the data mux and latch 718 are controlled by the outputs of control logic 720, and the values of the registers in the control logic 720 can be set by either SCP or CPU).*

As detailed above, claims 11-23 of the ‘576 patent are unpatentable over DeRoo under 35 U.S.C. 103(a).

**CONCLUSION**

Requester respectfully submits that, as detailed above, substantial new questions of patentability exist with respect to claims 1-23 of the '576 patent. Accordingly, Requester requests reexamination of claims 1-23 of the '576 patent. This Request presents overwhelming evidence that new issues of patentability exist with respect to claims 1-23 of the '576 patent, those claims should be rejected and revoked.

Respectfully submitted,  
J.C. PATENTS

Date: June 21, 2011

4 Venture, Suite 250  
Irvine, CA 92618  
Tel.: (949) 660-0761  
Fax: (949)-660-0809

/JIAWEI HUANG/  
Jiawei Huang  
Registration No. 43,330  
Email: [jcpi@msn.com](mailto:jcpi@msn.com)

|   |                                 |  |                    |
|---|---------------------------------|--|--------------------|
| FORM PTO-1449 U.S. DEPARTMENT OF COMMERCE<br>PATENT AND TRADEMARK OFFICE<br><b>INFORMATION DISCLOSURE STATEMENT BY<br/>         APPLICANT</b> | ATTY. DOCKET NO.: JCLA38862-RE  |  | APPLICATION No.: / |
|   | APPLICANT: IAN F. HARRIS et al. |  |                    |
|   | FILING DATE: HERewith           |  | GROUP:             |

#### U.S. PATENT DOCUMENTS

| EXAMINER<br>INITIAL |   | DOCUMENT NUMBER | DATE       | NAME                  | CLASS | SUBCLASS | FILING DATE<br>(IF APPROPRIATE) |
|---------------------|---|-----------------|------------|-----------------------|-------|----------|---------------------------------|
|                     | 1 | 6,161,162       | 12/12/2000 | David T. DeRoo et al. | 710   | 244      |                                 |
|                     |   |                 |            |                       |       |          |                                 |
|                     |   |                 |            |                       |       |          |                                 |
|                     |   |                 |            |                       |       |          |                                 |
|                     |   |                 |            |                       |       |          |                                 |
|                     |   |                 |            |                       |       |          |                                 |

#### FOREIGN PATENT DOCUMENTS

| EXAMINER<br>INITIAL |  | DOCUMENT NUMBER | PUBLICATION<br>DATE | COUNTRY | CLASS | SUBCLASS | ENGLISH<br>TRANSLATION |
|---------------------|--|-----------------|---------------------|---------|-------|----------|------------------------|
|                     |  |                 |                     |         |       |          |                        |
|                     |  |                 |                     |         |       |          |                        |
|                     |  |                 |                     |         |       |          |                        |

#### NON PATENT LITERATURE DOCUMENTS

| EXAMINER<br>INITIAL |  | Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or county where published. | ENGLISH<br>TRANSLATION |
|---------------------|--|--|------------------------|
|                     |  |  |                        |
|                     |  |  |                        |
|                     |  |  |                        |

|          |                 |
|----------|-----------------|
| EXAMINER | DATE CONSIDERED |
|----------|-----------------|

**EXAMINER:** INITIAL IF CITATION CONSIDERED, WHETHER OR NOT CITATION IS IN CONFORMANCE WITH MPEP 609; DRAW LINE THROUGH CITATION IF NOT IN CONFORMANCE AND NOT CONSIDERED. INCLUDE COPY OF THIS FORM WITH NEXT COMMUNICATION TO APPLICANT.

| Electronic Patent Application Fee Transmittal |          |  |        |                      |
|---|----------|--|--------|----------------------|
| Application Number:                           |          |  |        |                      |
| Filing Date:                                  |          |  |        |                      |
| Title of Invention:                           |          | SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM |        |                      |
| First Named Inventor/Applicant Name:          |          | IAN F. HARRIS  |        |                      |
| Filer:  |          | Jiawei Huang/Michelle Chang  |        |                      |
| Attorney Docket Number:                       |          | JCLA38862-RE   |        |                      |
| Filed as Large Entity                         |          |  |        |                      |
| <b>ex parte reexam Filing Fees</b>            |          |  |        |                      |
| Description                                   | Fee Code | Quantity   | Amount | Sub-Total in USD(\$) |
| <b>Basic Filing:</b>                          |          |  |        |                      |
| Request for ex parte reexamination            | 1812     | 1  | 2520   | 2520                 |
| <b>Pages:</b>                                 |          |  |        |                      |
| <b>Claims:</b>                                |          |  |        |                      |
| <b>Miscellaneous-Filing:</b>                  |          |  |        |                      |
| <b>Petition:</b>                              |          |  |        |                      |
| <b>Patent-Appeals-and-Interference:</b>       |          |  |        |                      |
| <b>Post-Allowance-and-Post-Issuance:</b>      |          |  |        |                      |
| <b>Extension-of-Time:</b>                     |          |  |        |                      |

| Description       | Fee Code | Quantity | Amount | Sub-Total in USD(\$) |
|-------------------|----------|----------|--------|----------------------|
| Miscellaneous:    |          |          |        |                      |
| Total in USD (\$) |          |          |        | 2520                 |



| Electronic Acknowledgement Receipt          |  |
|---|--|
| <b>EFS ID:</b>                              | 10355726   |
| <b>Application Number:</b>                  | 90011754   |
| <b>International Application Number:</b>    |  |
| <b>Confirmation Number:</b>                 | 4071   |
| <b>Title of Invention:</b>                  | SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM |
| <b>First Named Inventor/Applicant Name:</b> | IAN F. HARRIS  |
| <b>Customer Number:</b>                     | 23900  |
| <b>Filer:</b>                               | Jiawei Huang/Michelle Chang  |
| <b>Filer Authorized By:</b>                 | Jiawei Huang   |
| <b>Attorney Docket Number:</b>              | JCLA38862-RE   |
| <b>Receipt Date:</b>                        | 21-JUN-2011  |
| <b>Filing Date:</b>                         |  |
| <b>Time Stamp:</b>                          | 17:55:59   |
| <b>Application Type:</b>                    | Reexam (Third Party)   |

### Payment information:

|   |                 |
|---|-----------------|
| Submitted with Payment  | yes             |
| Payment Type  | Deposit Account |
| Payment was successfully received in RAM  | \$2520          |
| RAM confirmation Number   | 5030            |
| Deposit Account   | 500710          |
| Authorized User   |                 |
| <p>The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:</p> <p>Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)</p> |                 |

| File Listing:                |   |   |  |                  |                  |
|------------------------------|---|---|--|------------------|------------------|
| Document Number              | Document Description                                  | File Name                                   | File Size(Bytes)/ Message Digest         | Multi Part /.zip | Pages (if appl.) |
| 1                            | Transmittal of New Application                        | 38862-RE_Request_Ex_Parte_Reexamination.pdf | 421906                                   | no               | 3                |
|                              |   |   | 4bdc280e848016c266e6e6705c8e9cb9fa87a09f |                  |                  |
| Warnings:                    |   |   |  |                  |                  |
| Information:                 |   |   |  |                  |                  |
| 2                            | Copy of patent for which reexamination is requested   | 38862-RE_US7930576.pdf                      | 1183083                                  | no               | 15               |
|                              |   |   | 45ac3c31385a2452ecc01d7c39300891058ab81e |                  |                  |
| Warnings:                    |   |   |  |                  |                  |
| Information:                 |   |   |  |                  |                  |
| 3                            | Reexam - Affidavit/Decl/Exhibit Filed by 3rd Party    | 38862-RE_Reexamination_Statement.pdf        | 271873                                   | no               | 18               |
|                              |   |   | 4581cd394be9f3163c89adece5f56643113114a5 |                  |                  |
| Warnings:                    |   |   |  |                  |                  |
| Information:                 |   |   |  |                  |                  |
| 4                            | Reexam - Info Disclosure Statement Filed by 3rd Party | 38862-RE_IDS_PTO-1449.pdf                   | 30707                                    | no               | 1                |
|                              |   |   | 68318ce130a8c265302bb13b834b086cada70856 |                  |                  |
| Warnings:                    |   |   |  |                  |                  |
| Information:                 |   |   |  |                  |                  |
| 5                            | Reexam Miscellaneous Incoming Letter                  | 38862-RE_IDS_US6161162.pdf                  | 6490732                                  | no               | 143              |
|                              |   |   | f5239cc5038cbb4906b4e658367b3d5b44fe3442 |                  |                  |
| Warnings:                    |   |   |  |                  |                  |
| Information:                 |   |   |  |                  |                  |
| 6                            | Fee Worksheet (SB06)                                  | fee-info.pdf                                | 30323                                    | no               | 2                |
|                              |   |   | 08d7d19d7ba2ab5ab52aa1a50e5a9e2939faf2b7 |                  |                  |
| Warnings:                    |   |   |  |                  |                  |
| Information:                 |   |   |  |                  |                  |
| Total Files Size (in bytes): |   |   | 8428624                                  |                  |                  |

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

**New Applications Under 35 U.S.C. 111**

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

**National Stage of an International Application under 35 U.S.C. 371**

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

**New International Application Filed with the USPTO as a Receiving Office**

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

## Patent Assignment Abstract of Title

### Total Assignments: 1

Application #: 11958601

Filing Dt: 12/18/2007

Patent #: 7930576

Issue Dt: 04/19/2011

PCT #: NONE

Publication #: US20080256374

Pub Dt: 10/16/2008

Inventors: Ian F. Harris, Drew J. Dutton

Title: SHARING NON-SHARABLE DEVICES BETWEEN AN EMBEDDED CONTROLLER AND A PROCESSOR IN A COMPUTER SYSTEM

### Assignment: 1

Reel/Frame: 020262 / 0609

Received: 12/18/2007

Recorded: 12/18/2007

Mailed: 12/18/2007

Pages: 6

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignors: HARRIS, IAN F.

Exec Dt: 12/11/2007

DUTTON, DREW J.

Exec Dt: 12/05/2007

Assignee: STANDARD MICROSYSTEMS CORPORATION

80 ARKAY DRIVE

HAUPPAUGE, NEW YORK 11788

Correspondent: MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZE

P.O. BOX 398

AUSTIN, TX 78767-0398

Search Results as of: 06/24/2011 12:15 PM

---

If you have any comments or questions concerning the data displayed, contact PRD / Assignments at 571-272-3350. v.2.2  
Web interface last modified: Apr. 20, 2009