US009858084B2

(12) **United States Patent**
Manlapat et al.

(10) **Patent No.: US 9,858,084 B2**
(45) **Date of Patent: Jan. 2, 2018**

(54) **COPYING OF POWER-ON RESET SEQUENCER DESCRIPTOR FROM NONVOLATILE MEMORY TO RANDOM ACCESS MEMORY**

(71) Applicant: **BITMICRO Networks, Inc.**, Fremont, CA (US)

(72) Inventors: **Alvin Anonuevo Manlapat**, San Fernando (PH); **Ian Victor Pasion Beleno**, Quezon (PH)

(73) Assignee: **BITMICRO Networks, Inc.**, Fremont, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **14/217,399**

(22) Filed: **Mar. 17, 2014**

(65) **Prior Publication Data**

US 2014/0331034 A1 Nov. 6, 2014

Related U.S. Application Data

(60) Provisional application No. 61/801,952, filed on Mar. 15, 2013.

(51) **Int. Cl.**
G06F 1/12 (2006.01)
G06F 9/44 (2006.01)
G06F 1/24 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 9/4401*** (2013.01); ***G06F 1/12*** (2013.01); ***G06F 1/24*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 9/4401; G06F 1/12; G06F 1/24
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,752,871 A 6/1988 Sparks
5,111,058 A 5/1992 Martin
(Continued)

FOREIGN PATENT DOCUMENTS

JP 2005142859 A 6/2005
JP 2005-309847 11/2005
(Continued)

OTHER PUBLICATIONS

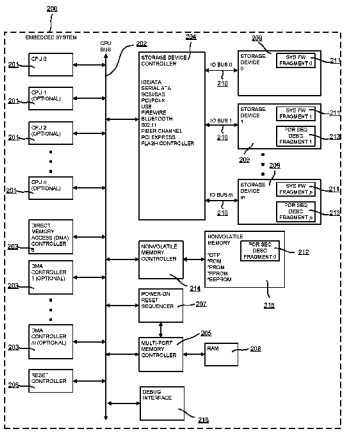
Notice of allowance/allowability for U.S. Appl. No. 14/217,041 dated Apr. 11, 2016.
(Continued)

Primary Examiner — Abdelmoniem Elamin

(57) **ABSTRACT**

A mechanism of booting up a system directly from a storage device and a means of initializing an embedded system prior to activating a CPU is presented. The said system is comprised of one or more CPUs, a reset controller, a storage device controller, one or more direct memory access controllers, a RAM and its controller, a ROM and its controller, a debug interface and a power-on reset (POR) sequencer. The POR sequencer controls the overall boot process of the embedded system. Said sequencer uses descriptors (POR Sequencer descriptors) which are used to update the configuration registers of the system and to enable CPU-independent data transfers with the use of DMA controllers. Using a minimal amount of non-volatile memory for booting up a system brings down costs associated with increased silicon real estate area and power consumption. Capability of pre-initializing the system even before a CPU is brought out of reset provides flexibility and system robustness. Through the use of the Power-On Reset Sequencer module, integrity of program code and user data used in the boot up process can be verified thus providing a resilient boot up sequence.

20 Claims, 9 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

RE34,100 E 10/1992 Hartness
5,222,046 A 6/1993 Kreifels et al.
5,297,148 A 3/1994 Harari et al.
5,341,339 A 8/1994 Wells
5,371,709 A 12/1994 Fisher et al.
5,379,401 A 1/1995 Robinson et al.
5,388,083 A 2/1995 Assar et al.
5,396,468 A 3/1995 Harari et al.
5,406,529 A 4/1995 Asano
5,432,748 A 7/1995 Hsu et al.
5,448,577 A 9/1995 Wells et al.
5,459,850 A 10/1995 Clay et al.
5,479,638 A 12/1995 Assar et al.
5,485,595 A 1/1996 Assar et al.
5,488,711 A 1/1996 Hewitt et al.
5,500,826 A 3/1996 Hsu et al.
5,509,134 A 4/1996 Fandrich et al.
5,513,138 A 4/1996 Manabe et al.
5,524,231 A 6/1996 Brown
5,530,828 A 6/1996 Kaki et al.
5,535,328 A 7/1996 Harari et al.
5,535,356 A 7/1996 Kim et al.
5,542,042 A 7/1996 Manson
5,542,082 A 7/1996 Solhjell
5,548,741 A 8/1996 Watanabe
5,559,956 A 9/1996 Sukegawa
5,568,423 A 10/1996 Jou et al.
5,568,439 A 10/1996 Harari
5,572,466 A 11/1996 Sukegawa
5,594,883 A 1/1997 Pricer
5,602,987 A 2/1997 Harari et al.
5,603,001 A 2/1997 Sukegawa et al.
5,606,529 A 2/1997 Honma et al.
5,606,532 A 2/1997 Lambrache et al.
5,619,470 A 4/1997 Fukumoto
5,627,783 A 5/1997 Miyauchi
5,640,349 A 6/1997 Kakinuma et al.
5,644,784 A 7/1997 Peek
5,682,509 A 10/1997 Kabenjian
5,737,742 A 4/1998 Achiwa et al.
5,787,466 A 7/1998 Berliner
5,796,182 A 8/1998 Martin
5,799,200 A 8/1998 Brant et al.
5,802,554 A 9/1998 Caceres et al.
5,819,307 A 10/1998 Iwamoto et al.
5,822,251 A 10/1998 Bruce et al.
5,875,351 A 2/1999 Riley
5,881,264 A 3/1999 Kurosawa
5,913,215 A 6/1999 Rubinstein et al.
5,918,033 A 6/1999 Heeb et al.
5,930,481 A 7/1999 Benhase
5,933,849 A 8/1999 Srbljic et al.
5,943,421 A 8/1999 Grabon
5,956,743 A 9/1999 Bruce et al.
6,000,006 A 12/1999 Bruce et al.
6,014,709 A 1/2000 Gulick et al.
6,076,137 A 6/2000 Asnaashari
6,098,119 A 8/2000 Surugucchi et al.
6,128,303 A 10/2000 Bergantino et al.
6,151,641 A 11/2000 Herbert
6,215,875 B1 4/2001 Nohda
6,230,269 B1 5/2001 Spies et al.
6,298,071 B1 10/2001 Taylor et al.
6,363,441 B1 3/2002 Beniz et al.
6,363,444 B1 3/2002 Platko et al.
6,397,267 B1 5/2002 Chong, Jr.
6,404,772 B1 6/2002 Beach et al.
6,496,939 B2 12/2002 Portman et al.
6,526,506 B1 2/2003 Lewis
6,529,416 B2 3/2003 Bruce et al.
6,557,095 B1 4/2003 Henstrom
6,601,126 B1 7/2003 Zaidi et al.
6,678,754 B1 1/2004 Soulier
6,744,635 B2 6/2004 Portman et al.
6,757,845 B2 6/2004 Bruce
6,857,076 B1 2/2005 Klein
6,901,499 B2 5/2005 Aasheim et al.
6,922,391 B1 7/2005 King et al.
6,961,805 B2 11/2005 Lakhani et al.
6,970,446 B2 11/2005 Krischer et al.
6,970,890 B1 11/2005 Bruce et al.
6,973,546 B2 12/2005 Johnson
6,980,795 B1 12/2005 Hermann et al.
7,103,684 B2 * 9/2006 Chen G06F 13/28 710/20
7,174,438 B2 2/2007 Homma et al.
7,194,766 B2 3/2007 Noehring et al.
7,263,006 B2 8/2007 Aritome
7,283,629 B2 10/2007 Kaler et al.
7,305,548 B2 12/2007 Pierce et al.
7,330,954 B2 2/2008 Nangle
7,372,962 B2 5/2008 Fujimoto et al.
7,386,662 B1 6/2008 Kekre et al.
7,412,631 B2 8/2008 Uddenberg et al.
7,415,549 B2 8/2008 Vemula et al.
7,424,553 B1 9/2008 Borrelli et al.
7,430,650 B1 9/2008 Ross
7,474,926 B1 1/2009 Carr et al.
7,490,177 B2 2/2009 Kao
7,500,063 B2 3/2009 Zohar et al.
7,506,098 B2 3/2009 Arcedera et al.
7,613,876 B2 11/2009 Bruce et al.
7,620,748 B1 11/2009 Bruce et al.
7,624,239 B2 11/2009 Bennett et al.
7,636,801 B1 12/2009 Kekre et al.
7,660,941 B2 2/2010 Lee et al.
7,668,925 B1 2/2010 Liao et al.
7,676,640 B2 3/2010 Chow et al.
7,681,188 B1 3/2010 Tirumalai et al.
7,716,389 B1 5/2010 Bruce et al.
7,719,287 B2 5/2010 Marks et al.
7,729,370 B1 6/2010 Orcine et al.
7,743,202 B2 6/2010 Tsai et al.
7,765,359 B2 7/2010 Kang et al.
7,877,639 B2 1/2011 Hoang
7,913,073 B2 3/2011 Choi
7,921,237 B1 * 4/2011 Holland G06F 11/1004 710/22
7,934,052 B2 4/2011 Prins et al.
7,958,295 B1 6/2011 Liao et al.
7,979,614 B1 7/2011 Yang
8,010,740 B2 8/2011 Arcedera et al.
8,032,700 B2 10/2011 Bruce et al.
8,156,279 B2 4/2012 Tanaka et al.
8,156,320 B2 4/2012 Borras et al.
8,161,223 B1 4/2012 Chamseddine et al.
8,165,301 B1 4/2012 Bruce et al.
8,200,879 B1 6/2012 Falik et al.
8,219,719 B1 7/2012 Parry et al.
8,225,022 B2 7/2012 Caulkins
8,341,311 B1 12/2012 Szewerenko et al.
8,375,257 B2 2/2013 Hong et al.
8,447,908 B2 5/2013 Bruce et al.
8,510,631 B2 8/2013 Wu et al.
8,560,804 B2 10/2013 Bruce et al.
8,707,134 B2 4/2014 Takahashi et al.
8,713,417 B2 4/2014 Jo
8,762,609 B1 6/2014 Lam et al.
8,788,725 B2 7/2014 Bruce et al.
8,832,371 B2 9/2014 Uehara et al.
8,856,392 B2 10/2014 Myrah et al.
8,959,307 B1 2/2015 Bruce et al.
9,043,669 B1 5/2015 Bruce et al.
9,099,187 B2 8/2015 Bruce et al.
9,135,190 B1 9/2015 Bruce et al.
9,147,500 B2 9/2015 Kim et al.
2001/0010066 A1 7/2001 Chin et al.
2002/0044486 A1 4/2002 Chan et al.
2002/0073324 A1 6/2002 Hsu et al.
2002/0083262 A1 6/2002 Fukuzumi
2002/0083264 A1 6/2002 Coulson
2002/0141244 A1 10/2002 Bruce et al.
2003/0023817 A1 1/2003 Rowlands et al.
2003/0065836 A1 4/2003 Pecone

(56) **References Cited**

U.S. PATENT DOCUMENTS

2003/0120864 A1 6/2003 Lee et al.
2003/0126451 A1 7/2003 Gorobets
2003/0131201 A1 7/2003 Khare et al.
2003/0161355 A1 8/2003 Falcomato et al.
2003/0163624 A1 8/2003 Matsui et al.
2003/0163647 A1 8/2003 Cameron et al.
2003/0163649 A1 8/2003 Kapur et al.
2003/0182576 A1 9/2003 Morlang et al.
2003/0188100 A1 10/2003 Solomon et al.
2003/0204675 A1 10/2003 Dover et al.
2003/0217202 A1 11/2003 Zilberman et al.
2003/0223585 A1 12/2003 Tardo et al.
2004/0073721 A1 4/2004 Goff et al.
2004/0128553 A1 7/2004 Buer et al.
2005/0050245 A1 3/2005 Miller et al.
2005/0078016 A1 4/2005 Neff
2005/0097368 A1 5/2005 Peinado et al.
2005/0120146 A1* 6/2005 Chen G06F 13/28 710/22
2005/0210149 A1 9/2005 Kimball
2005/0210159 A1 9/2005 Voorhees et al.
2005/0226407 A1 10/2005 Kasuya et al.
2005/0243610 A1 11/2005 Guha et al.
2005/0289361 A1 12/2005 Sutardja
2006/0004957 A1 1/2006 Hand, III et al.
2006/0031450 A1 2/2006 Unrau et al.
2006/0039406 A1 2/2006 Day et al.
2006/0095709 A1 5/2006 Achiwa
2006/0112251 A1 5/2006 Karr et al.
2006/0184723 A1 8/2006 Sinclair et al.
2007/0019573 A1 1/2007 Nishimura
2007/0028040 A1 2/2007 Sinclair
2007/0058478 A1 3/2007 Murayama
2007/0073922 A1 3/2007 Go et al.
2007/0079017 A1 4/2007 Brink et al.
2007/0083680 A1 4/2007 King et al.
2007/0088864 A1 4/2007 Foster
2007/0093124 A1 4/2007 Varney et al.
2007/0094450 A1 4/2007 VanderWiel
2007/0096785 A1 5/2007 Maeda
2007/0121499 A1 5/2007 Pal et al.
2007/0130439 A1 6/2007 Andersson et al.
2007/0159885 A1 7/2007 Gorobets
2007/0168754 A1 7/2007 Zohar et al.
2007/0174493 A1 7/2007 Irish et al.
2007/0174506 A1 7/2007 Tsuruta
2007/0195957 A1 8/2007 Arulambalam et al.
2007/0288686 A1 12/2007 Arcedera et al.
2007/0288692 A1 12/2007 Bruce et al.
2007/0294572 A1 12/2007 Kalwitz et al.
2008/0052456 A1 2/2008 Ash et al.
2008/0072031 A1 3/2008 Choi
2008/0104264 A1 5/2008 Duerk et al.
2008/0147963 A1 6/2008 Tsai et al.
2008/0189466 A1 8/2008 Hemmi
2008/0195800 A1 8/2008 Lee et al.
2008/0218230 A1 9/2008 Shim
2008/0228959 A1 9/2008 Wang
2008/0276037 A1 11/2008 Chang et al.
2009/0028229 A1 1/2009 Cagno et al.
2009/0037565 A1 2/2009 Andresen et al.
2009/0055573 A1 2/2009 Ito
2009/0077306 A1 3/2009 Arcedera et al.
2009/0083022 A1 3/2009 Nordin et al.
2009/0094411 A1 4/2009 Que
2009/0132620 A1 5/2009 Arakawa
2009/0132752 A1 5/2009 Poo et al.
2009/0150643 A1 6/2009 Jones et al.
2009/0158085 A1 6/2009 Kern et al.
2009/0172250 A1 7/2009 Allen et al.
2009/0172466 A1 7/2009 Royer et al.
2009/0240873 A1 9/2009 Yu et al.
2010/0058045 A1 3/2010 Borras et al.
2010/0095053 A1 4/2010 Bruce et al.
2010/0125695 A1 5/2010 Wu et al.
2010/0250806 A1 9/2010 Devilla et al.
2010/0268904 A1 10/2010 Sheffield et al.
2010/0299538 A1 11/2010 Miller
2011/0022778 A1 1/2011 Schibilla et al.
2011/0022783 A1 1/2011 Moshayedi
2011/0022801 A1 1/2011 Flynn
2011/0087833 A1 4/2011 Jones
2011/0093648 A1 4/2011 Belluomini et al.
2011/0113186 A1 5/2011 Bruce et al.
2011/0145479 A1 6/2011 Talagala et al.
2011/0161568 A1 6/2011 Bruce et al.
2011/0167204 A1 7/2011 Estakhri et al.
2011/0197011 A1 8/2011 Suzuki et al.
2011/0202709 A1 8/2011 Rychlik
2011/0208914 A1 8/2011 Winokur et al.
2011/0219150 A1 9/2011 Piccirillo et al.
2011/0258405 A1 10/2011 Asaki et al.
2011/0264884 A1 10/2011 Kim
2011/0264949 A1 10/2011 Ikeuchi et al.
2011/0270979 A1 11/2011 Schlansker et al.
2012/0005405 A1 1/2012 Wu et al.
2012/0005410 A1 1/2012 Ikeuchi
2012/0017037 A1 1/2012 Riddle et al.
2012/0079352 A1 3/2012 Frost et al.
2012/0102263 A1 4/2012 Aswadhati
2012/0102268 A1 4/2012 Smith et al.
2012/0137050 A1 5/2012 Wang et al.
2012/0161568 A1 6/2012 Umemoto et al.
2012/0260102 A1 10/2012 Zaks et al.
2012/0271967 A1 10/2012 Hirschman
2012/0303924 A1 11/2012 Ross
2012/0311197 A1 12/2012 Larson et al.
2012/0324277 A1 12/2012 Weston-Lewis et al.
2013/0010058 A1 1/2013 Pomeroy
2013/0019053 A1 1/2013 Somanache et al.
2013/0073821 A1 3/2013 Flynn et al.
2013/0094312 A1 4/2013 Jang et al.
2013/0099838 A1 4/2013 Kim et al.
2013/0111135 A1 5/2013 Bell, Jr. et al.
2013/0208546 A1 8/2013 Kim et al.
2013/0212337 A1 8/2013 Maruyama
2013/0212349 A1 8/2013 Maruyama
2013/0246694 A1 9/2013 Bruce et al.
2013/0254435 A1 9/2013 Shapiro et al.
2013/0262750 A1 10/2013 Yamasaki et al.
2013/0282933 A1 10/2013 Jokinen et al.
2013/0304775 A1 11/2013 Davis et al.
2013/0339578 A1 12/2013 Ohya et al.
2013/0339582 A1 12/2013 Olbrich et al.
2013/0346672 A1 12/2013 Sengupta et al.
2014/0068177 A1 3/2014 Raghavan
2014/0095803 A1 4/2014 Kim et al.
2014/0104949 A1 4/2014 Bruce et al.
2014/0108869 A1 4/2014 Brewerton et al.
2014/0189203 A1 7/2014 Suzuki et al.
2014/0258788 A1 9/2014 Maruyama
2014/0285211 A1 9/2014 Raffinan
2014/0331034 A1 11/2014 Ponce et al.
2015/0006766 A1 1/2015 Ponce et al.
2015/0012690 A1 1/2015 Bruce et al.
2015/0032937 A1 1/2015 Salessi
2015/0032938 A1 1/2015 Salessi
2015/0067243 A1 3/2015 Salessi et al.
2015/0149697 A1 5/2015 Salessi et al.
2015/0149706 A1 5/2015 Salessi et al.
2015/0153962 A1 6/2015 Salessi et al.
2015/0169021 A1 6/2015 Salessi et al.
2015/0261456 A1 9/2015 Alcantara et al.
2015/0261475 A1 9/2015 Alcantara et al.
2015/0261797 A1 9/2015 Alcantara et al.
2015/0370670 A1 12/2015 Lu
2015/0371684 A1 12/2015 Mataya
2015/0378932 A1 12/2015 Souri et al.
2016/0026402 A1 1/2016 Alcantara et al.
2016/0027521 A1 1/2016 Lu
2016/0041596 A1 2/2016 Alcantara et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

FOREIGN PATENT DOCUMENTS

TW	489308	6/2002
TW	200428219 A	12/2004
TW	436689	12/2005
TW	I420316	12/2013
WO	WO 94/06210	3/1994
WO	WO 98/38568	9/1998

OTHER PUBLICATIONS

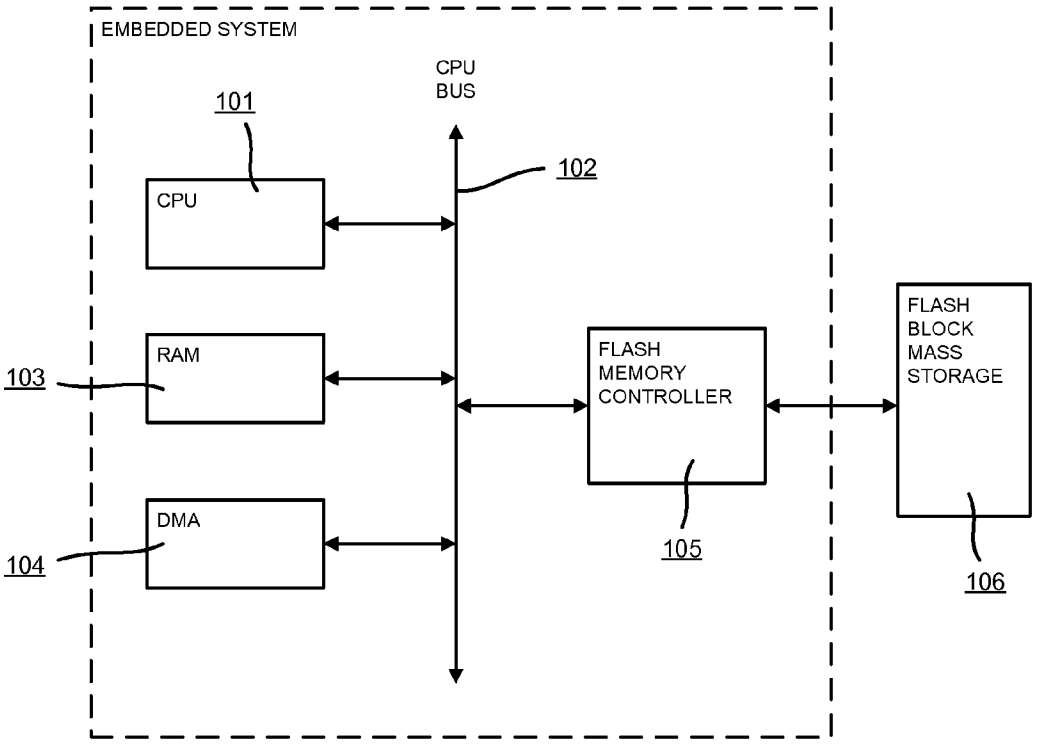
Office Action for U.S. Appl. No. 13/475,878, mailed on Jun. 23, 2014.
Office Action for U.S. Appl. No. 13/253,912 mailed on Jul. 16, 2014.
Office Action for U.S. Appl. No. 12/876,113 mailed on Jul. 11, 2014.
Office Action for U.S. Appl. No. 12/270,626 mailed on Feb. 3, 2012.
Office Action for U.S. Appl. No. 12/270,626 mailed on Apr. 4, 2011.
Office Action for U.S. Appl. No. 12/270,626 mailed on Mar. 15, 2013.
Notice of Allowance/Allowability for U.S. Appl. No. 12/270,626 mailed on Oct. 3, 2014.
Advisory Action for U.S. Appl. No. 12/876,113 mailed on Oct. 16, 2014.
Office Action for U.S. Appl. No. 14/297,628 mailed on Jul. 17, 2015.
Office Action for U.S. Appl. No. 13/475,878 mailed on Dec. 4, 2014.
Office Action mailed Sep. 11, 2015 for U.S. Appl. No. 14/217,436.
Office Action mailed Sep. 24, 2015 for U.S. Appl. No. 14/217,334.
Office Action dated Sep. 18, 2015 for Taiwanese Patent Application No. 102144165.
Office Action mailed Sep. 29, 2015 for U.S. Appl. No. 14/217,316.
Office Action mailed Sep. 28, 2015 for U.S. Appl. No. 14/689,045.
Office Action for U.S. Appl. No. 14/217,365 dated Feb. 18, 2016.
Office Action for U.S. Appl. No. 14/217,365 dated Mar. 2, 2016.
Office Action for U.S. Appl. No. 14/690,305 dated Feb. 25, 2016.
Office Action for U.S. Appl. No. 14/217,436 dated Feb. 25, 2016.
Office Action for U.S. Appl. No. 14/217,316 dated Feb. 26, 2016.
Office Action for U.S. Appl. No. 14/215,414 dated Mar. 1, 2016.
Office Action for U.S. Appl. No. 14/616,700 dated Mar. 8, 2016.
Notice of allowance/allowability for U.S. Appl. No. 13/253,912 dated Mar. 21, 2016.
Notice of allowance/allowability for U.S. Appl. No. 14/803,107 dated Mar. 28, 2016.
Office Action for U.S. Appl. No. 14/217,334 dated Apr. 4, 2016.
Notice of allowance/allowability for U.S. Appl. No. 14/214,216 dated Apr. 27, 2016.
Notice of allowance/allowability for U.S. Appl. No. 14/217,436 dated May 6, 2016.
Advisory Action for U.S. Appl. No. 14/217,334 dated Jun. 13, 2016.
Office Action for U.S. Appl. No. 14/217,291 dated Jun. 15, 2016.
USPTO Notice of Allowability & attachment(s) mailed Jan. 7, 2013 for U.S. Appl. No. 12/876,247.
Office Action mailed Sep. 14, 2012 for U.S. Appl. No. 12/876,247.
Office Action mailed Feb. 1, 2012 for U.S. Appl. No. 12/876,247.
Notice of Allowance/Allowability mailed Mar. 31, 2015 for U.S. Appl. No. 13/475,878.
Office Action mailed May 22, 2015 for U.S. Appl. No. 13/253,912.
Office Action for U.S. Appl. No. 13/475,878 mailed on Jun. 23, 2014.
Office Action for U.S. Appl. No. 12/876,113 mailed on Oct. 16, 2014.
Notice of Allowance for U.S. Appl. No. 12/270,626 mailed Oct. 3, 2014.
Office Action for U.S. Appl. No. 12/270,626 mailed on May 23, 2014.
Office Action for U.S. Appl. No. 12/270,626 mailed on Dec. 18, 2013.
Office Action for U.S. Appl. No. 12/270,626 mailed on Aug. 23, 2012.
Office Action for U.S. Appl. No. 14/217,249 dated Apr. 21, 2016.
Notice of allowance/allowability for U.S. Appl. No. 14/217,467 dated Apr. 20, 2016.
Notice of allowance/allowability for U.S. Appl. No. 14/214,216 dated.
Office Action for U.S. Appl. No. 12/876,113 mailed on Mar. 13, 2014.
Advisory Action for U.S. Appl. No. 12/876,113 mailed on Sep. 6, 2013.
Office Action for U.S. Appl. No. 12/876,113 mailed on May 14, 2013.
Office Action for U.S. Appl. No. 12/876,113 mailed on Dec. 21, 2012.
William Stallings, Security Comes to SNMP: The New SNMPv3 Proposed Internet Standard, The Internet Protocol Journal, vol. 1, No. 3, Dec. 1998.
Notice of Allowability for U.S. Appl. No. 12/882,059 mailed on May 30, 2013.
Notice of Allowability for U.S. Appl. No. 12/882,059 mailed on Feb. 14, 2013.
Office Action for U.S. Appl. No. 12/882,059 mailed on May 11, 2012.
Notice of Allowability for U.S. Appl. No. 14/038,684 mailed on Aug. 1, 2014.
Office Action for U.S. Appl. No. 14/038,684 mailed on Mar. 17, 2014.
Office Action mailed Oct. 8, 2015 for U.S. Appl. No. 14/217,291.
Final Office Action mailed Nov. 19, 2015 for U.S. Appl. No. 14/217,249.
Final Office Action mailed Nov. 18, 2015 for U.S. Appl. No. 14/217,467.
Office Action mailed Nov. 25, 2015 for U.S. Appl. No. 14/217,041.
Office Action mailed Oct. 5, 2015 for Taiwanese Application No. 103105076.
Office Action mailed Nov. 19, 2015 for U.S. Appl. No. 14/217,249.
Office Action mailed Nov. 18, 2015 for U.S. Appl. No. 14/217,467.
Office Action mailed Dec. 4, 2015 for U.S. Appl. No. 14/616,700.
Office Action mailed Jun. 4, 2015 for U.S. Appl. No. 14/215,414.
Notice of Allowance/Allowability for U.S. Appl. No. 13/890,229 mailed on Feb. 20, 2014.
Office Action for U.S. Appl. No. 13/890,229 mailed on Oct. 8, 2013.
Office Action for U.S. Appl. No. 12/876,113 mailed on Dec. 5, 2014.
Notice of Allowance/Allowabilty for U.S. Appl. No. 12/876,113 mailed on Jun. 22, 2015.
Office Action for U.S. Appl. No. 14/217,249 mailed on Apr. 23, 2015.
Office Action for U.S. Appl. No. 14/217,467 mailed on Apr. 27, 2015.
Office Action for U.S. Appl. No. 14/616,700 mailed on Apr. 30, 2015.
Office Action for U.S. Appl. No. 14/217,436 mailed on Sep. 11, 2015.
Office Action mailed Dec. 15, 2015 for U.S. Appl. No. 13/253,912.
Office Action mailed Dec. 17, 2015 for U.S. Appl. No. 14/214,216.
Office Action mailed Dec. 17, 2015 for U.S. Appl. No. 14/215,414.
Office Action mailed Dec. 17, 2015 for U.S. Appl. No. 14/803,107.
Office Action mailed Jan. 15, 2016 for U.S. Appl. No. 14/866,946.
Office Action mailed Jan. 11, 2016 for U.S. Appl. No. 14/217,399.
Office Action mailed Jan. 15, 2016 for U.S. Appl. No. 14/216,937.
Notice of Allowance and Examiner-Initiated Interview Summary, mailed Jan. 29, 2016 for U.S. Appl. No. 14/297,628.
Office Action for U.S. Appl. No. 14/215,414 dated May 20, 2016.
Office Action for U.S. Appl. No. 14/616,700 dated May 20, 2016.
Office Action for U.S. Appl. No. 14/689,019 dated May 20, 2016.
Advisory Action for U.S. Appl. No. 14/217,316 dated May 19, 2016.
National Science Fountation,Award Abstract #1548968, SBIR Phase I: SSD In-Situ Processing, http://www.nsf.gov/awardsearch/showAward?AWD_ID=1548968 printed on Feb. 13, 2016.

(56) **References Cited**

OTHER PUBLICATIONS

Design-Reuse, NxGn Data Emerges from Stealth Mode to provide a paradigm shift in enterprise storage solution (author(s) not indicated).
http://www.design-reuse.com/news/35111/nxgn-data-intelligent-solutions.html, printed on Feb. 13, 2016 (author(s) not indicated).
Office Action mailed Dec. 5, 2014 for U.S. Appl. No. 14/038,684.
Office Action for U.S. Appl. No. 14/217,096 dated Jul. 12, 2016.
Notice of Allowance for U.S. Appl. No. 14/217,399 dated Jul. 20, 2016.
Office Action for U.S. Appl. No. 14/866,946 dated Jul. 29, 2016.
Notice of Allowance for U.S. Appl. No. 14/217,334 dated Jul. 29, 2016.
Office Action for U.S. Appl. No. 14/690,243 dated Aug. 11, 2016.
Office Action for U.S. Appl. No. 14/690,370 dated Aug. 12, 2016.
Office Action for U.S. Appl. No. 14/216,937 dated Aug. 15, 2016.
Working Draft American National Standard Project T10/1601-D Information Technology Serial Attached SCSI—1.1 (SAS—1.1), Mar. 13, 2004 Revision 4.
Office Action for U.S. Appl. No. 14/217,316 dated Aug. 25, 2016.
Office Action for U.S. Appl. No. 14/690,305 dated Aug. 26, 2016.
Advisory Action for U.S. Appl. No. 14/217,291 dated Sep. 9, 2016.
Advisory Action for U.S. Appl. No. 14/689,045 dated Sep. 16, 2016.
Notice of Allowance for U.S. Appl. No. 14/182,303 dated Sep. 12, 2016.
Advisory Action for U.S. Appl. No. 14/690,114 dated Sep. 12, 2016.
Notice of Allowance for U.S. Appl. No. 14/215,414 dated Sep. 23, 2016.
Advisory Action for U.S. Appl. No. 14/866,946 dated Oct. 13, 2016.
Office Action for U.S. Appl. No. 14/687,700 dated Sep. 26, 2016.
Office Action for U.S. Appl. No. 15/170,768 dated Oct. 6, 2016.
Notice of allowance/allowability for U.S. Appl. No. 14/217,365 dated Oct. 18, 2016.
Office Action for U.S. Appl. No. 14/684,399 dated May 12, 2017.
Notice of allowance/allowability for U.S. Appl. No. 14/217,399 dated May 12, 2017.
Notice of allowance/allowability for U.S. Appl. No. 14/217,365 dated Jul. 21, 2017.

* cited by examiner



PRIOR ART

FIG. 1

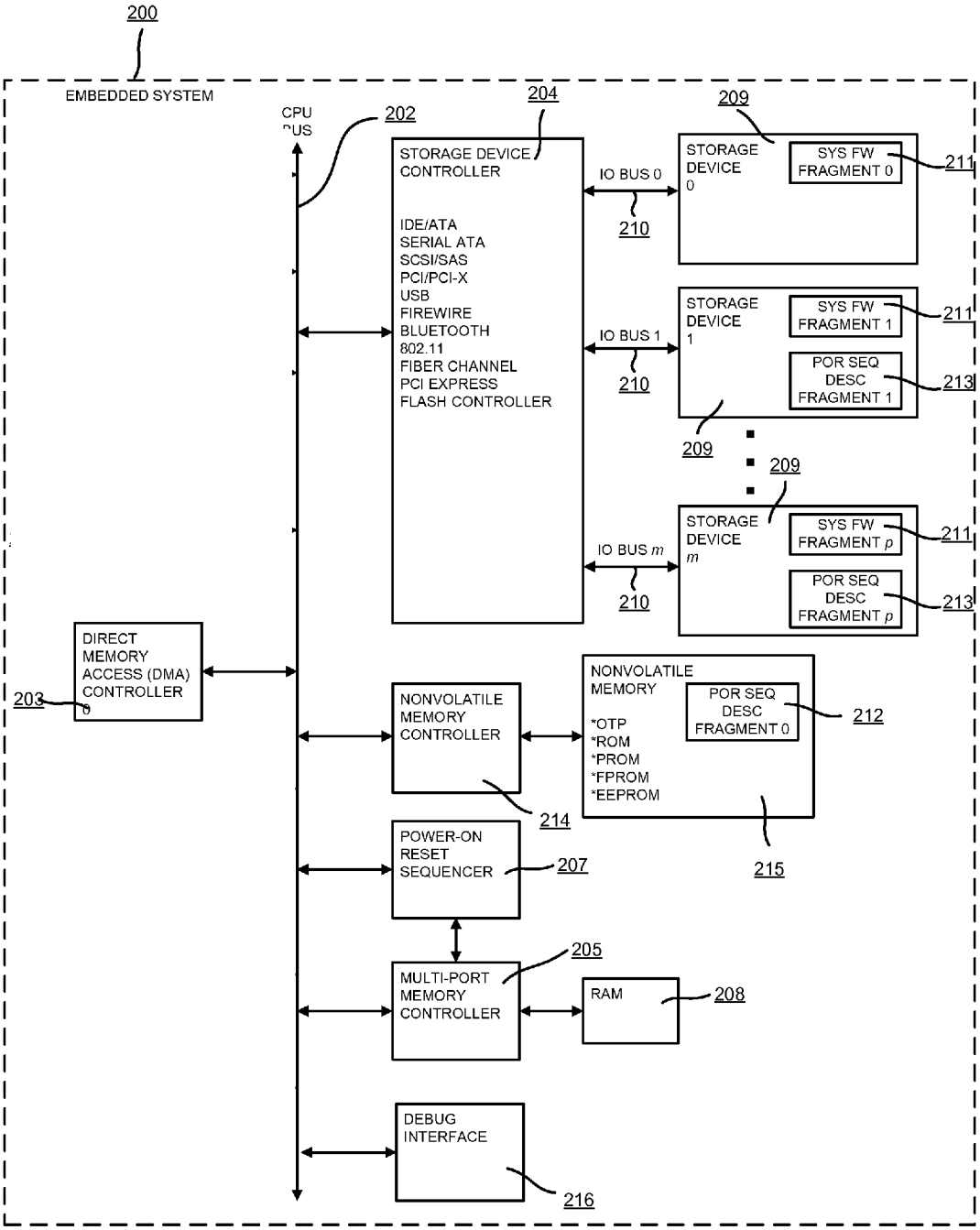

200
EMBEDDED SYSTEM
CPU BUS
202
204
STORAGE DEVICE CONTROLLER
IDE/ATA
SERIAL ATA
SCSI/SAS
PCI/PCI-X
USB
FIREWIRE
BLUETOOTH
802.11
FIBER CHANNEL
PCI EXPRESS
FLASH CONTROLLER
IO BUS 0
IO BUS 1
IO BUS m
210
209
STORAGE DEVICE 0
STORAGE DEVICE 1
STORAGE DEVICE m
SYS FW FRAGMENT 0
SYS FW FRAGMENT 1
SYS FW FRAGMENT p
POR SEQ DESC FRAGMENT 1
POR SEQ DESC FRAGMENT p
211
213
DIRECT MEMORY ACCESS (DMA) CONTROLLER 0
203
NONVOLATILE MEMORY CONTROLLER
214
NONVOLATILE MEMORY
*OTP
*ROM
*PROM
*FPROM
*EEPROM
POR SEQ DESC FRAGMENT 0
212
215
POWER-ON RESET SEQUENCER
207
MULTI-PORT MEMORY CONTROLLER
205
RAM
208
DEBUG INTERFACE
216

FIG. 2

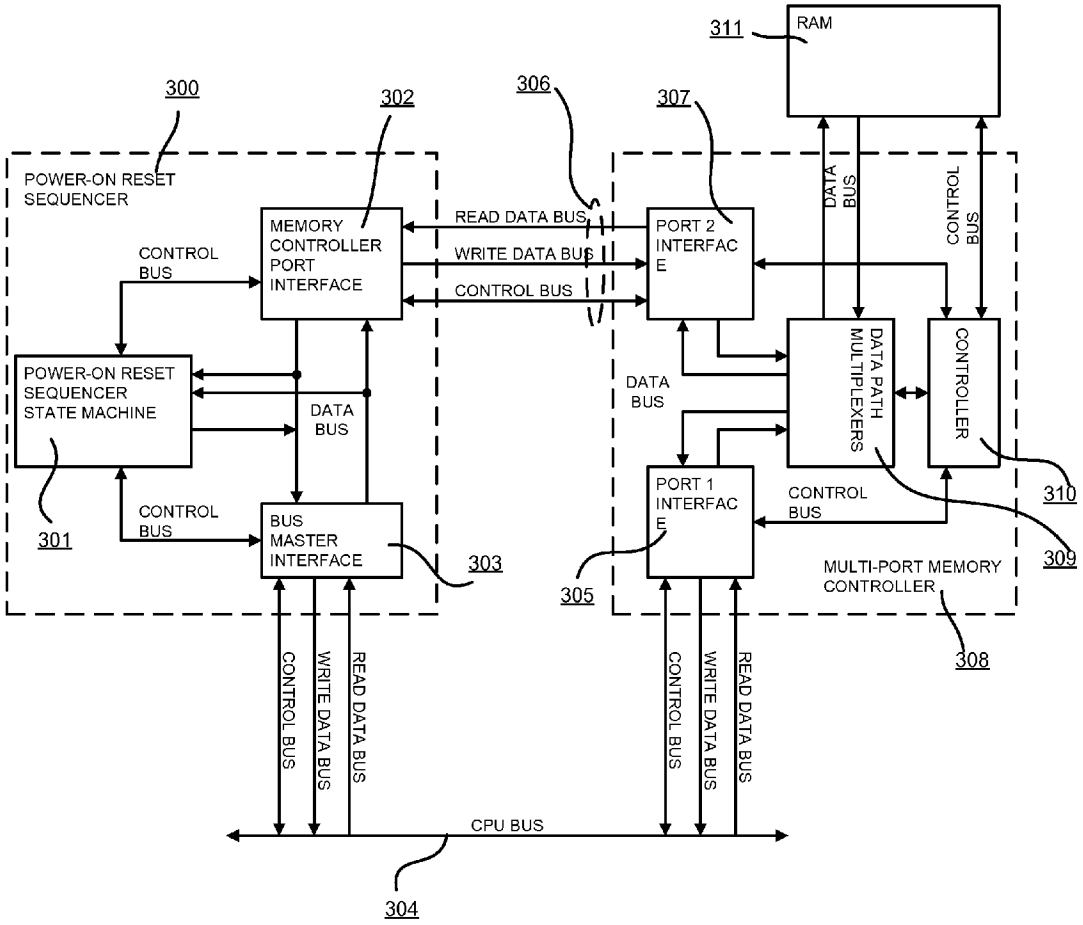

311
RAM
300
302
306
307
POWER-ON RESET SEQUENCER
MEMORY CONTROLLER PORT INTERFACE
READ DATA BUS
WRITE DATA BUS
CONTROL BUS
PORT 2 INTERFACE
DATA BUS
CONTROL BUS
CONTROL BUS
POWER-ON RESET SEQUENCER STATE MACHINE
DATA BUS
DATA BUS
DATA PATH MULTIPLEXERS
CONTROLLER
310
301
CONTROL BUS
BUS MASTER INTERFACE
303
PORT 1 INTERFACE
CONTROL BUS
309
305
MULTI-PORT MEMORY CONTROLLER
308
CONTROL BUS
WRITE DATA BUS
READ DATA BUS
CONTROL BUS
WRITE DATA BUS
READ DATA BUS
CPU BUS
304

FIG. 3

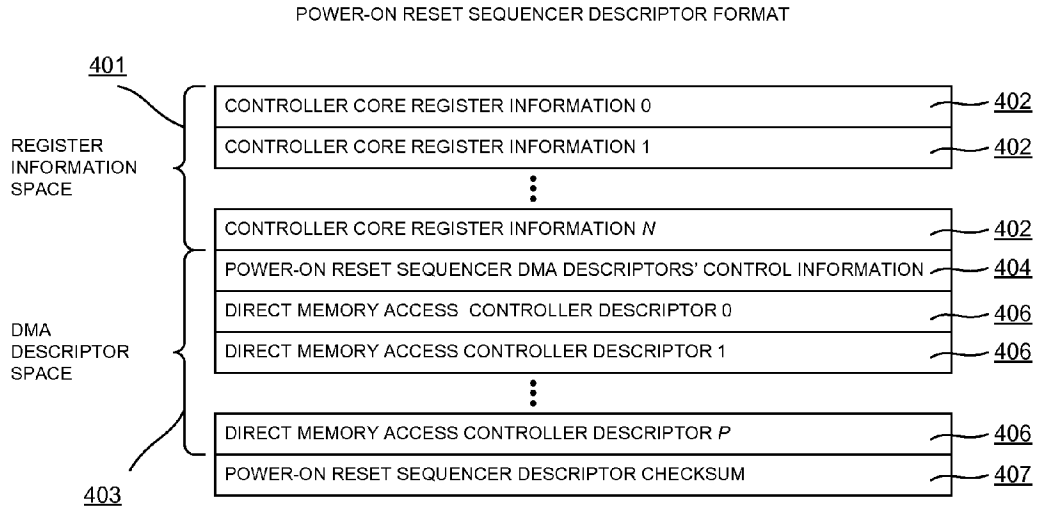

POWER-ON RESET SEQUENCER DESCRIPTOR FORMAT
401
REGISTER INFORMATION SPACE
DMA DESCRIPTOR SPACE
403
CONTROLLER CORE REGISTER INFORMATION 0
402
CONTROLLER CORE REGISTER INFORMATION 1
402
CONTROLLER CORE REGISTER INFORMATION N
402
POWER-ON RESET SEQUENCER DMA DESCRIPTORS' CONTROL INFORMATION
404
DIRECT MEMORY ACCESS CONTROLLER DESCRIPTOR 0
406
DIRECT MEMORY ACCESS CONTROLLER DESCRIPTOR 1
406
DIRECT MEMORY ACCESS CONTROLLER DESCRIPTOR P
406
POWER-ON RESET SEQUENCER DESCRIPTOR CHECKSUM
407

FIG. 4

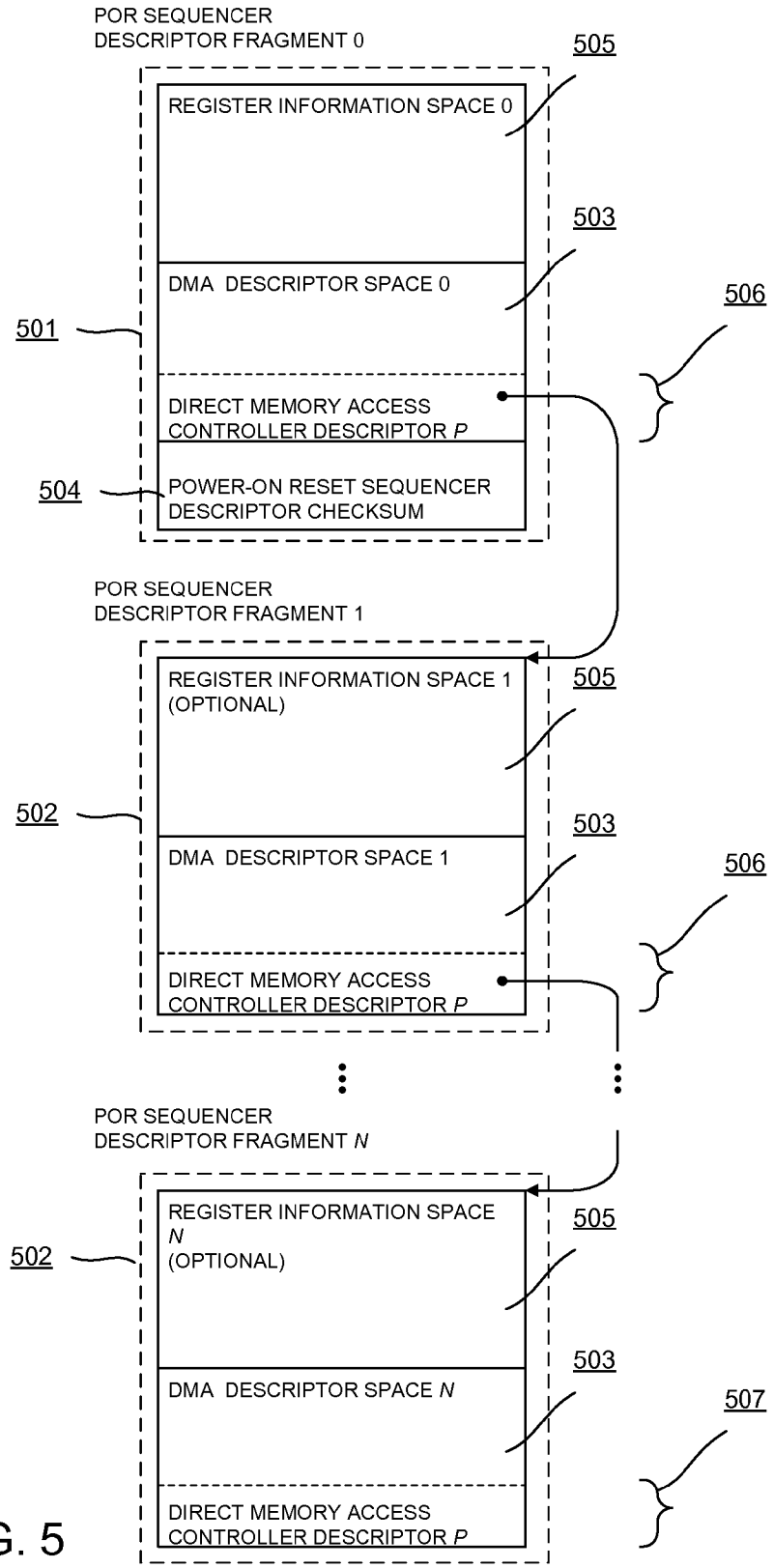

POR SEQUENCER
DESCRIPTOR FRAGMENT 0
505
REGISTER INFORMATION SPACE 0
503
DMA DESCRIPTOR SPACE 0
506
501
DIRECT MEMORY ACCESS
CONTROLLER DESCRIPTOR P
504
POWER-ON RESET SEQUENCER
DESCRIPTOR CHECKSUM
POR SEQUENCER
DESCRIPTOR FRAGMENT 1
REGISTER INFORMATION SPACE 1
(OPTIONAL)
505
502
503
DMA DESCRIPTOR SPACE 1
506
DIRECT MEMORY ACCESS
CONTROLLER DESCRIPTOR P
POR SEQUENCER
DESCRIPTOR FRAGMENT N
REGISTER INFORMATION SPACE
N
(OPTIONAL)
505
502
503
DMA DESCRIPTOR SPACE N
507
DIRECT MEMORY ACCESS
CONTROLLER DESCRIPTOR P

FIG. 5

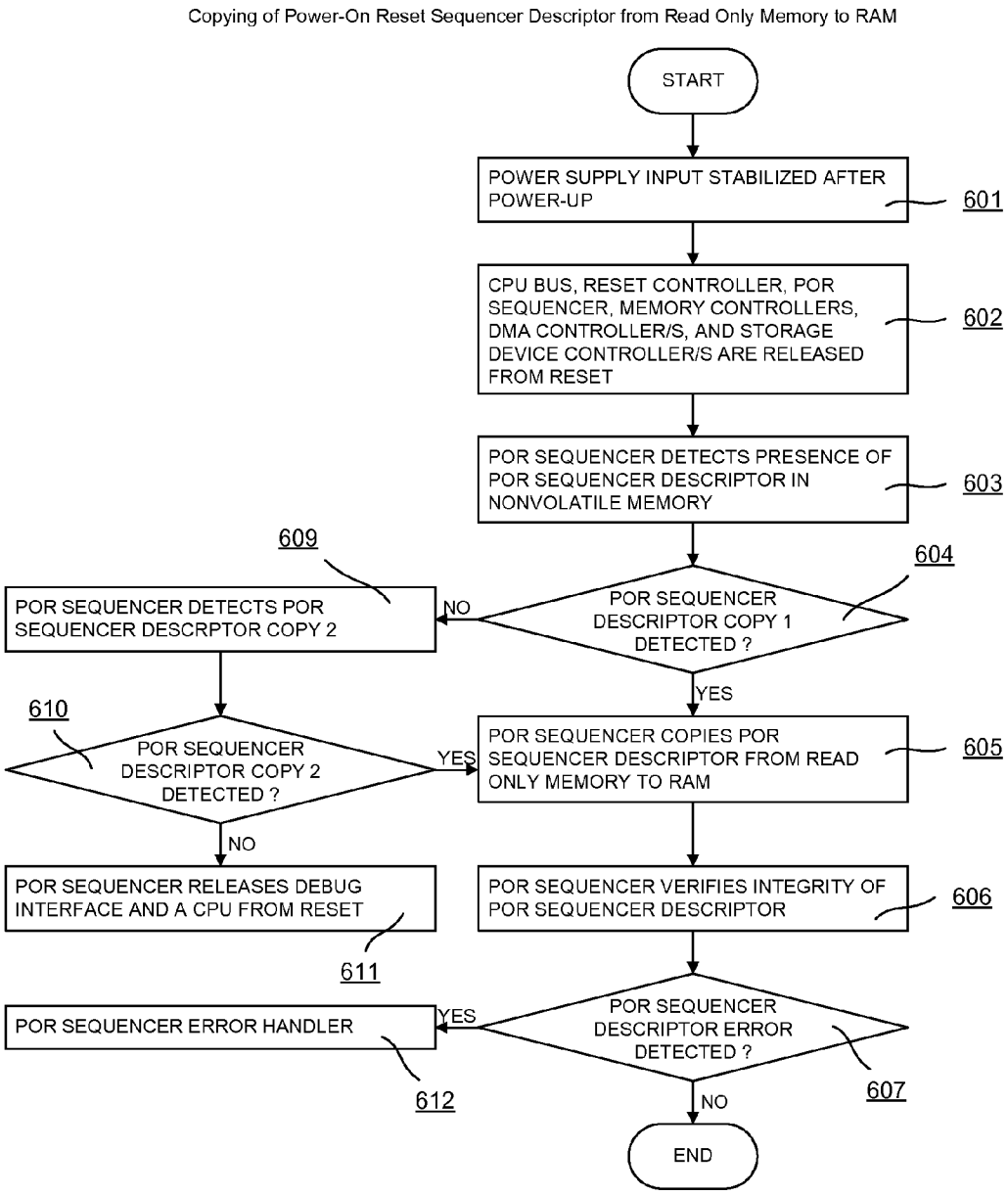

Copying of Power-On Reset Sequencer Descriptor from Read Only Memory to RAM
START
POWER SUPPLY INPUT STABILIZED AFTER POWER-UP
601
CPU BUS, RESET CONTROLLER, POR SEQUENCER, MEMORY CONTROLLERS, DMA CONTROLLER/S, AND STORAGE DEVICE CONTROLLER/S ARE RELEASED FROM RESET
602
POR SEQUENCER DETECTS PRESENCE OF POR SEQUENCER DESCRIPTOR IN NONVOLATILE MEMORY
603
609
604
POR SEQUENCER DETECTS POR SEQUENCER DESCRPTOR COPY 2
NO
POR SEQUENCER DESCRIPTOR COPY 1 DETECTED ?
YES
610
POR SEQUENCER DESCRIPTOR COPY 2 DETECTED ?
YES
POR SEQUENCER COPIES POR SEQUENCER DESCRIPTOR FROM READ ONLY MEMORY TO RAM
605
NO
POR SEQUENCER RELEASES DEBUG INTERFACE AND A CPU FROM RESET
POR SEQUENCER VERIFIES INTEGRITY OF POR SEQUENCER DESCRIPTOR
606
611
POR SEQUENCER ERROR HANDLER
YES
POR SEQUENCER DESCRIPTOR ERROR DETECTED ?
612
607
NO
END

FIG. 6

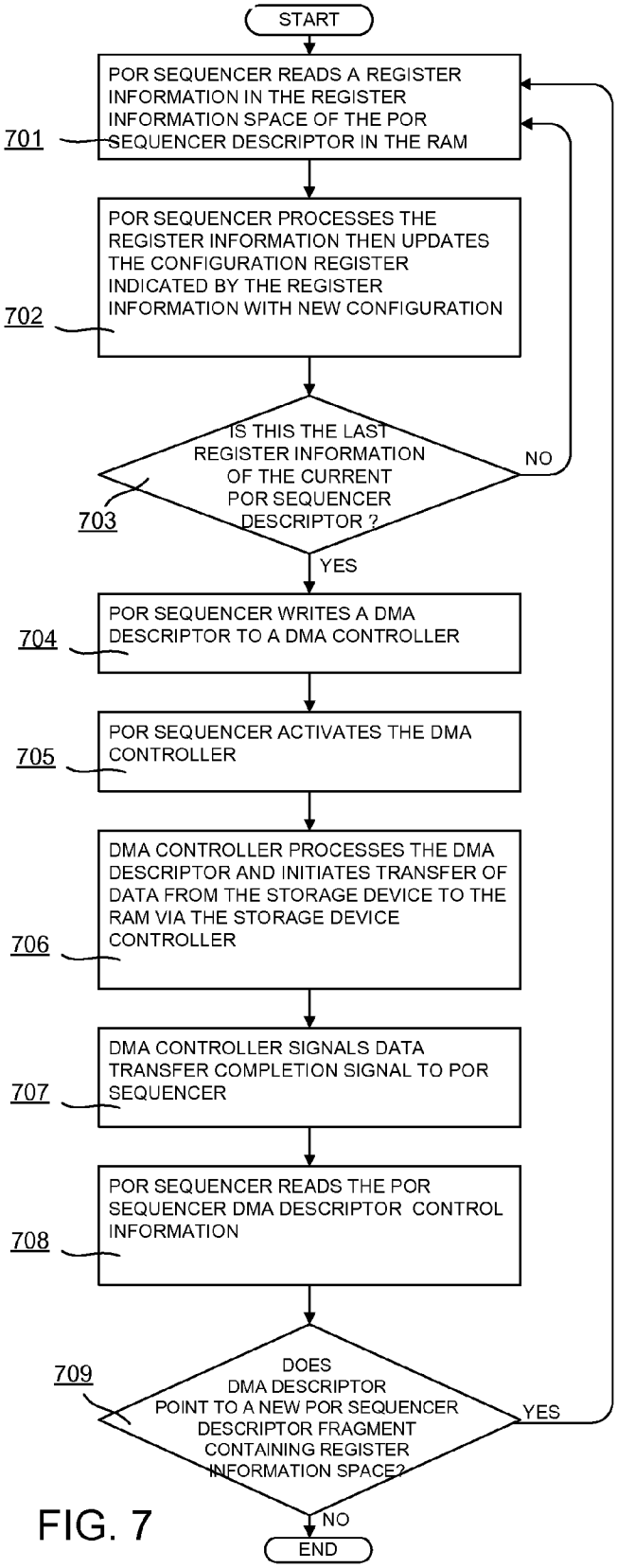

START
POR SEQUENCER READS A REGISTER INFORMATION IN THE REGISTER INFORMATION SPACE OF THE POR SEQUENCER DESCRIPTOR IN THE RAM
701
POR SEQUENCER PROCESSES THE REGISTER INFORMATION THEN UPDATES THE CONFIGURATION REGISTER INDICATED BY THE REGISTER INFORMATION WITH NEW CONFIGURATION
702
IS THIS THE LAST REGISTER INFORMATION OF THE CURRENT POR SEQUENCER DESCRIPTOR ?
703
NO
YES
POR SEQUENCER WRITES A DMA DESCRIPTOR TO A DMA CONTROLLER
704
POR SEQUENCER ACTIVATES THE DMA CONTROLLER
705
DMA CONTROLLER PROCESSES THE DMA DESCRIPTOR AND INITIATES TRANSFER OF DATA FROM THE STORAGE DEVICE TO THE RAM VIA THE STORAGE DEVICE CONTROLLER
706
DMA CONTROLLER SIGNALS DATA TRANSFER COMPLETION SIGNAL TO POR SEQUENCER
707
POR SEQUENCER READS THE POR SEQUENCER DMA DESCRIPTOR CONTROL INFORMATION
708
DOES DMA DESCRIPTOR POINT TO A NEW POR SEQUENCER DESCRIPTOR FRAGMENT CONTAINING REGISTER INFORMATION SPACE?
709
YES
NO
END

FIG. 7

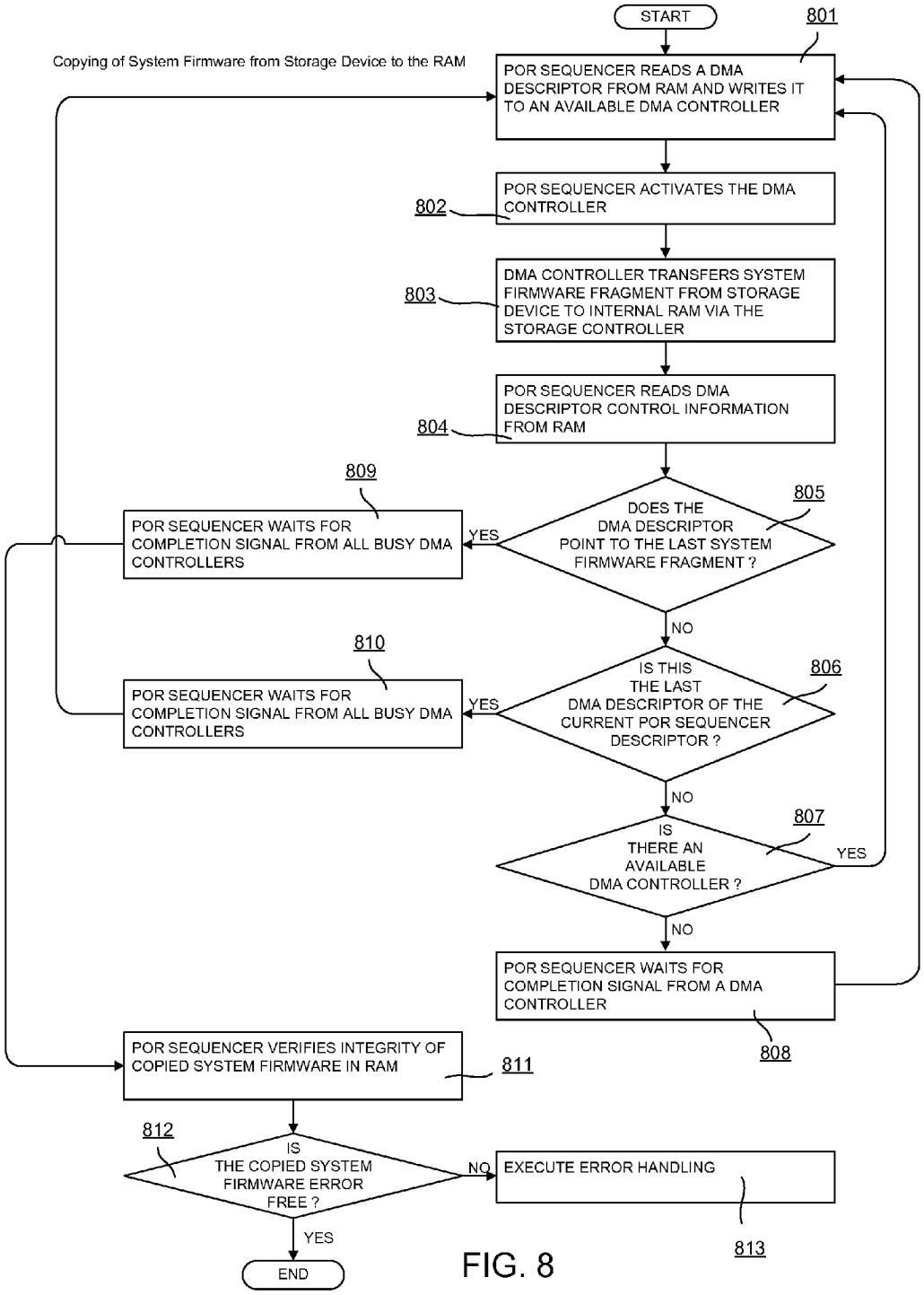

START
801
Copying of System Firmware from Storage Device to the RAM
POR SEQUENCER READS A DMA DESCRIPTOR FROM RAM AND WRITES IT TO AN AVAILABLE DMA CONTROLLER
POR SEQUENCER ACTIVATES THE DMA CONTROLLER
802
DMA CONTROLLER TRANSFERS SYSTEM FIRMWARE FRAGMENT FROM STORAGE DEVICE TO INTERNAL RAM VIA THE STORAGE CONTROLLER
803
POR SEQUENCER READS DMA DESCRIPTOR CONTROL INFORMATION FROM RAM
804
809
805
DOES THE DMA DESCRIPTOR POINT TO THE LAST SYSTEM FIRMWARE FRAGMENT ?
YES
POR SEQUENCER WAITS FOR COMPLETION SIGNAL FROM ALL BUSY DMA CONTROLLERS
NO
810
IS THIS THE LAST DMA DESCRIPTOR OF THE CURRENT POR SEQUENCER DESCRIPTOR ?
806
YES
POR SEQUENCER WAITS FOR COMPLETION SIGNAL FROM ALL BUSY DMA CONTROLLERS
NO
807
IS THERE AN AVAILABLE DMA CONTROLLER ?
YES
NO
POR SEQUENCER WAITS FOR COMPLETION SIGNAL FROM A DMA CONTROLLER
808
POR SEQUENCER VERIFIES INTEGRITY OF COPIED SYSTEM FIRMWARE IN RAM
811
812
IS THE COPIED SYSTEM FIRMWARE ERROR FREE ?
NO
EXECUTE ERROR HANDLING
813
YES
END

FIG. 8

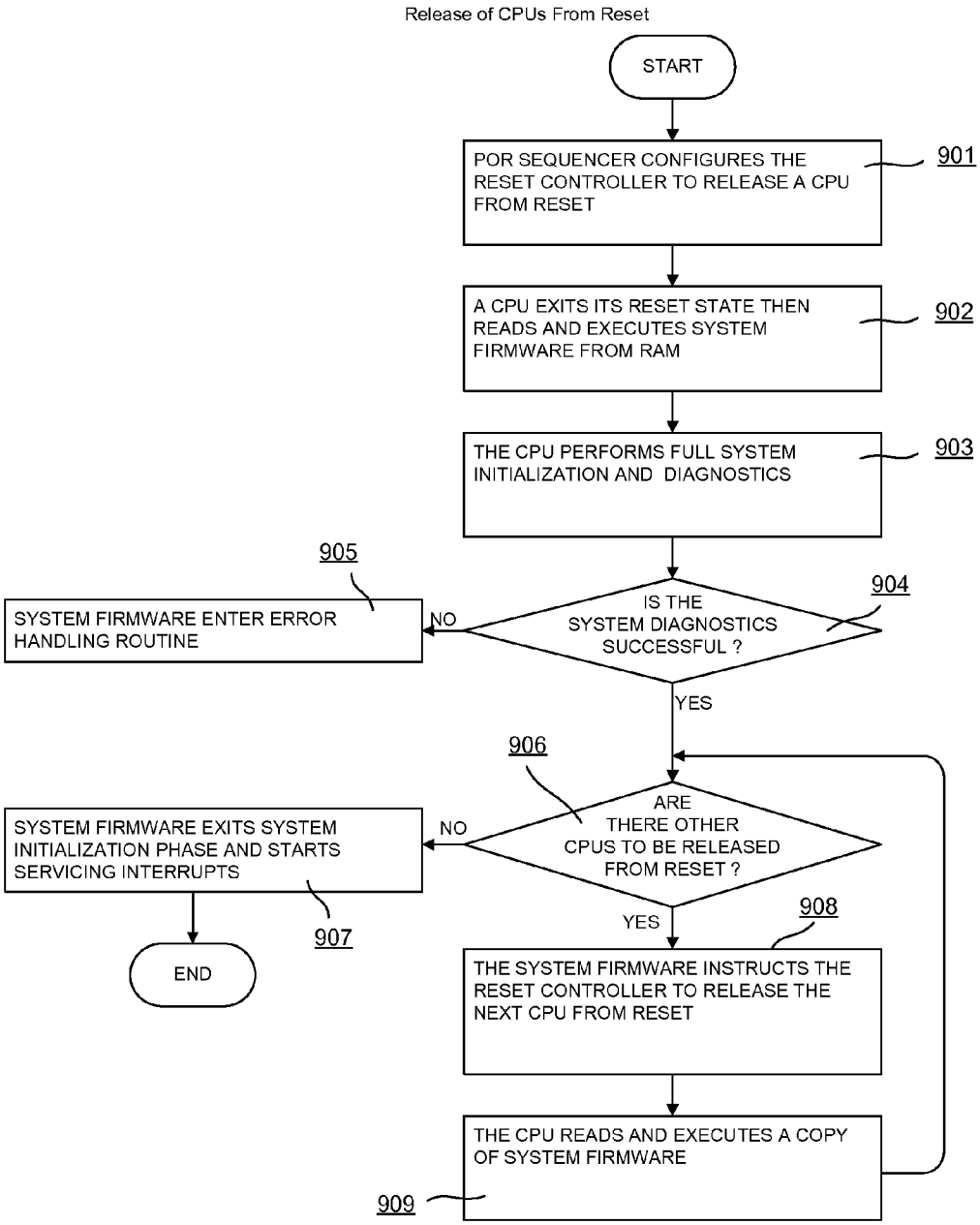

Release of CPUs From Reset
START
POR SEQUENCER CONFIGURES THE RESET CONTROLLER TO RELEASE A CPU FROM RESET
901
A CPU EXITS ITS RESET STATE THEN READS AND EXECUTES SYSTEM FIRMWARE FROM RAM
902
THE CPU PERFORMS FULL SYSTEM INITIALIZATION AND DIAGNOSTICS
903
905
IS THE SYSTEM DIAGNOSTICS SUCCESSFUL ?
904
NO
SYSTEM FIRMWARE ENTER ERROR HANDLING ROUTINE
YES
906
ARE THERE OTHER CPUS TO BE RELEASED FROM RESET ?
NO
SYSTEM FIRMWARE EXITS SYSTEM INITIALIZATION PHASE AND STARTS SERVICING INTERRUPTS
907
END
YES
908
THE SYSTEM FIRMWARE INSTRUCTS THE RESET CONTROLLER TO RELEASE THE NEXT CPU FROM RESET
THE CPU READS AND EXECUTES A COPY OF SYSTEM FIRMWARE
909

FIG. 9

COPYING OF POWER-ON RESET SEQUENCER DESCRIPTOR FROM NONVOLATILE MEMORY TO RANDOM ACCESS MEMORY

CROSS-REFERENCE(S) TO RELATED APPLICATIONS

This application claims the benefit of and priority to U.S. Provisional Application 61/801,952, filed 15 Mar. 2013. This U.S. Provisional Application 61/801,952 is hereby fully incorporated herein by reference.

BACKGROUND

Field

The present invention relates generally to computers having a single or multiple Central Processing Units (CPU), and more particularly, to a method and device for booting such a system directly from a storage device.

Description of Related Art

An embedded system refers to an apparatus that is made up of a single or multiple controller chips interconnected to each other on a printed circuit board. A controller chip can have one or more central processing unit (CPU) that enables it to interpret and execute code. When an embedded system has more than one processor it is already called a multiprocessor system. The term embedded system will be used to refer to a single CPU or a multiprocessor system in this paper.

A conventional embedded system contains a nonvolatile memory which is used to store firmware that is read and executed by a CPU to initialize the system. The nonvolatile memory can be a read-only memory (ROM), one-time programmable memory (OTP), programmable read only memory (PROM), electrically erasable PROM (EEPROM), or a flash PROM (FPROM). The nonvolatile memory can be internal (on-chip) or external (off-chip) to the controller chip. The firmware stored in the nonvolatile memory can be a boot code loader, a basic input/output system (BIOS), an operating system (OS), or an application firmware. The boot code loader is a small code executed by a CPU to do minimum initialization of the system. After initialization, the boot code loader copies the BIOS or OS or application firmware from a mass storage device external to the embedded system to the random access memory (RAM) of the system. When the BIOS or OS or application firmware is already loaded in the RAM, the CPU reads and executes it to do exhaustive initialization and diagnostics before proceeding to normal system operation. A mass storage device can be a rotating disk drive or a solid-state drive which is made up of FPROM devices. Though nonvolatile memory size can be reduced due to a small boot code loader, the boot code loader doesn't have the ability to self check its integrity.

In an invention described in US Patent Application Publication number US 2005/0120146 A1, Jun. 2 2005, an embedded system in FIG. **1** that does not have an on-chip nonvolatile memory such as a read only memory directly boots from flash block mass storage **106**. The flash block mass storage **106** is a collection of flash chips interfaced to the flash memory controller **105**. The apparatus makes use of a Flash Programming Engine, embedded within a Flash memory controller **105**, the first device that is taken out of reset, to program a Direct Memory Access controller **104** to transfer a boot code loader from flash block mass storage **106** to an internal Random Access Memory (RAM) **103**.

When the transfer of the boot code loader to the RAM **103** is completed the internal Central Processing Unit **101** of the controller chip is taken out of reset to read and execute the boot code loader in the RAM **103**. The boot code loader performs initialization of the system, power-on self tests, and copies a much larger control program or application firmware from flash block mass storage **106** to the RAM **103**.

SUMMARY

The invention that is described in this document makes use of a minimum amount of non-volatile memory in order to boot an embedded system. Reduction in size of the nonvolatile memory brings about an advantage in terms of cost and complexity. The invention also allows the system to verify the integrity of data and code used in the boot process.

In some cases, the configuration registers of an embedded system may need to be initialized even before any of the CPU of the system is brought out of reset to begin execution of the system firmware. The invention presented in this paper will also address this issue of pre-initialization.

The present invention provides a mechanism for booting up a system using a minimum amount of nonvolatile memory. This method also enables the embedded system to initialize all configuration registers even before any of the CPUs of the system is brought out of reset. The embedded system consists of multiple controller chips or a single controller chip. The embedded system can have a single or multiple central processing units.

The present invention comprises of at least one Central Processing Unit (CPU) which reads and executes firmware; a random access memory (RAM) which is used to temporarily store firmware; a memory controller which provides interface to the RAM; a Reset Controller which maintains the reset states of all the devices of the embedded system; a Power-On Reset Sequencer which controls the overall boot process of the embedded system; nonvolatile memory and controller used to store data used by the Power-On Reset Sequencer in the boot process; at least one direct memory access (DMA) controller which initiates and tracks data transfers; a storage device controller which provides interface to the storage device; a CPU bus which interconnects the devices of the system; and a storage device which is used to store system firmware; a debug interface used to preprogram the nonvolatile memory and storage device with data and code.

The present invention describes the structure of the Power-On Reset (POR) Sequencer and its operation. The POR Sequencer makes use of a POR Sequencer Descriptor which is a preassembled descriptor that is stored in a nonvolatile memory of the system. The POR Sequencer descriptor contains register information and direct memory access (DMA) controller descriptors. The register information are read and used by the POR Sequencer to update the configuration registers of the system. The DMA descriptors are loaded by the POR Sequencer to the DMA controllers of the system. The POR Sequencer facilitates the boot process and takes care of verifying the integrity of loaded data and code in the RAM.

The POR Sequencer Descriptor of the current invention is not limited to a single data structure. A linked-list of data structures is presented to increase the amount of register information and DMA descriptors available for processing by the POR Sequencer.

The present invention also provides a method of copying POR Sequencer Descriptor from nonvolatile memory to the

RAM and verifying its integrity; a method of initializing the configuration registers by the POR Sequencer; a method of copying the system firmware from the storage device to the RAM; and a method of releasing the CPUs of the system from reset.

Initially, the system's nonvolatile memory and storage device contain nothing. When the system is powered up, no boot process occurs but the debug interface and a CPU are released from reset. The system integrator loads a program to the RAM via the debug interface. The program contains code to burn the POR Sequencer Descriptor in nonvolatile memory and to save a copy of the System Firmware in the storage device.

The method of copying POR Sequencer Descriptor from nonvolatile memory to RAM starts when the supply voltage of the system becomes stable after power-up. Then the Reset Controller, the POR Sequencer, the CPU Bus, the memory controllers, the DMA controllers, and the storage device controllers are released from reset. Then the POR Sequencer detects the presence of the POR Sequencer Descriptor in the nonvolatile memory. If the POR Sequencer Descriptor is not detected, then the debug interface and a CPU of the system are released from reset. If the POR Sequencer Descriptor is detected, then the POR Sequencer copies the POR Sequencer Descriptor from nonvolatile memory to RAM. This step ends after the verification of the copied POR Sequencer Descriptor to the RAM completed without error.

The method of initializing the configuration registers of the system starts when the first register information of the POR Sequencer Descriptor is read. Then the register indicated by the register information is updated with a new set of configuration. This sequence is repeated until the last register information of the POR Sequencer Descriptor is read and processed. Then the step proceeds to the processing of the DMA descriptor. The type of the DMA descriptor is identified by the POR Sequencer. If the DMA descriptor points to another set of register information then the DMA transfer will update the register information and DMA descriptor spaces of the current POR Sequence Descriptor in the RAM. If the DMA descriptor points to a system firmware fragment then this step is ended and the sequence proceeds to the next method.

The next method is the step where the system firmware stored in the external storage device is copied to the RAM of the embedded system. The system firmware can be a boot code loader, a BIOS, an operating system, or a plain application firmware. This step starts when the POR Sequencer identifies that the DMA descriptor points to a system firmware fragment. The POR Sequencer loads the DMA or DMAs with the DMA Descriptors that will copy the system firmware fragments to the RAM. The POR Sequencer waits for completion signals from all DMA controllers that have been loaded with DMA descriptors before proceeding in the process flow. If the last DMA descriptor of the current POR Sequencer Descriptor points to another set of DMA Descriptors then another set of system firmware fragments must be copied from the storage devices to the RAM. If the last DMA descriptor of the current POR Sequencer Descriptor points to the last system firmware fragment then this step ends. The storage device refers to a single storage device or a plurality of storage devices of the same type.

The last step is the release of the CPUs of the embedded system from reset. When the system has a single CPU, after the system firmware is loaded in the RAM, the POR Sequencer configures the reset controller to release the CPU from reset. Then the CPU starts reading and executing the system firmware. In a multi-CPU system, a CPU is released from reset. Then, the CPU reads and executes the system firmware in the RAM. The system firmware will have instructions to sequence the release of the other CPUs from reset either one at a time or simultaneously.

BRIEF DESCRIPTION OF DRAWINGS

So that the manner in which the above recited features, advantages and objects of the present invention are attained and can be understood in detail, a more particular description of the invention, briefly summarized above, may be had by reference to the embodiments thereof which are illustrated in the appended drawings.

It is to be noted, however, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the present invention may admit to other equally effective embodiments.

FIG. **1** shows a prior art embedded system that implements a mechanism to directly boot from flash block mass storage.

FIG. **2** is a block diagram of an embedded system with a power-on reset sequencer that manages the initialization and boot up of the system.

FIG. **3** is a block diagram which presents the internal organization of the power-on reset sequencer and its interconnection to the RAM via a memory controller.

FIG. **4** shows the structure of the power-on reset sequencer descriptor.

FIG. **5** illustrates a linked-list data structure of the power-on reset sequencer descriptor.

FIG. **6** is a flow chart of the sequence used in copying the power-on reset sequencer descriptor from nonvolatile memory to the RAM.

FIG. **7** is a flow chart of the sequence used in the register initialization phase of the power-on reset sequencer.

FIG. **8** is a flow chart of the sequence used in copying the system firmware from the storage device to the RAM.

FIG. **9** is a flow chart of the sequence used in releasing the CPUs of the embedded system from their reset states.

DETAILED DESCRIPTION

The present invention relates a method and device for booting up a system from a storage device and further relates a method for enabling a system to initialize all configuration registers even before any of the CPUs of the system is brought out of reset.

FIG. **2** is a block diagram showing a typical embodiment of the present invention. The figure shows an embedded system **200** comprising a CPU bus **202** interconnecting a single CPU **201** or a plurality of CPUs **201**, a single DMA controller **203** or a plurality of DMA controllers **203**, a reset controller **206**, a power-on reset (POR) sequencer **207**, a RAM controller **205**, a nonvolatile memory controller **214**, a storage device controller **204**, and a debug interface **216**. Also comprising the system is an internal RAM **208**, a small nonvolatile memory **215** and a storage device **209**. A storage device type may be a mechanical disk drive, a mobile flash drive, or a solid-state drive.

The storage device controller **204** controls the storage device or devices **209** connected to it. A storage device or a plurality of storage devices **209** are connected to the storage device controller **204** via an IO bus **210** for the case of interface standards such as IDE/ATA, Serial ATA, SCSI/SAS, PCI/PCI-X, USB, Firewire, Bluetooth, Fibre Channel and PCI Express. For a solid-state drive, a flash bus is used

to connect to the storage device controller **204** which also called a flash controller. The reset controller **206** handles the reset states of all the devices in the system. One or a plurality of DMA controllers **203** initiates and tracks data transfers. The storage device controller **204** provides interface to the storage device **209**. The storage device **209** is used to store system firmware **211** or a fragment of the system firmware **211**. When the POR sequencer descriptor exceeds the space of the nonvolatile memory, the descriptor is divided into fragments **212**, **213** and distributed across the nonvolatile memory **215** and storage devices **209**.

The multi-port memory controller **205** interfaces the RAM **208** to the CPU bus **202** and to the POR sequencer **207**. The POR sequencer **207** processes the POR sequencer descriptor which contains information to initialize configuration registers of the system and DMA descriptors used to fetch other POR sequencer descriptor fragments **213** and the system firmware **211**. The POR sequencer **207** takes care of loading the system firmware to the RAM and verifying its integrity. After the system firmware has been loaded in the RAM and the system's configuration registers pre-initialized, the POR sequencer sequences the release of the CPUs from their reset states.

The nonvolatile memory **215** with its controller **214** is used to store the head of the POR sequencer descriptor **212** used by the POR sequencer **207** in the boot process. Initially, the system's nonvolatile memory and storage devices attached to the system are empty. When the system is powered up the boot process exits to emulation mode since there are no POR sequencer descriptor to process and no system firmware to load and execute in memory. The system integrator which writes program for the system loads a firmware to the RAM of the system via the debug interface **216**. The firmware loaded in the RAM has a routine that writes the POR sequencer descriptor and system firmware to the nonvolatile memory **215** and storage devices **209** attached to the system. After the POR sequencer descriptor and system firmware has been written to the nonvolatile memory and storage devices, the next time the system is powered up, the system doesn't exit to emulation mode instead it continues with the boot process until all CPUs **201** of the system are released from their reset states.

FIG. **3** is a schematic diagram that illustrates the structure and interconnection between the POR sequencer **300** and the multi-port memory controller **308**. For the POR sequencer **300**, data and control signals to and from the CPU bus **304** is managed by the bus master interface **303** while control signals to and from the RAM **311** is managed by the memory controller port interface **302**. The POR sequencer state machine **301** utilizes the bus master interface **303** to perform data transfer to and from the CPU bus and the memory controller port interface **302** to do data transfer to and from the RAM **311**. The POR sequencer **300** combines the use of the bus master interface **303** and the memory controller port interface **302** in order to control transfers to and from the RAM **311**.

The POR sequencer **300** transfers the POR sequencer descriptor from the nonvolatile memory or from the storage device to the RAM **311** before it is processed. When the POR sequencer descriptor is already loaded in the RAM **311**, the POR sequencer state machine reads a part of the descriptor from RAM to parse and interpret. Depending on the content of the part of the POR sequencer descriptor, the POR sequencer state machine decides whether to get another part of the descriptor from RAM, transfer data from RAM to a device mapped on the CPU bus **304**, or to write data that it generated to the RAM or to a device mapped on the CPU bus **304**. Through the use of POR sequencer state machine **301**, memory controller port interface **302**, and bus master interface **303** the POR sequencer **300** is able to do the following tasks: transfer of DMA controller descriptor to the DMA controller's configuration register space, activation of the DMA controller, initialization of configuration registers, and copy of data from storage device or nonvolatile memory to the RAM.

The multi-port memory controller **308** has several port interfaces. Port **1** interface **305** provides devices on the CPU bus **304** access to the RAM **311**. Port **2** interface **307** provides the POR sequencer **300** access to the RAM **311**. The data path multiplexers **309** route the read and write requests from the POR sequencer **300** and the CPU bus **304** to the RAM **311**. The controller **310** manages how the data path multiplexers **309** behave in response to the read and write requests received from the POR sequencer **300** and CPU bus **304**.

FIG. **4** shows the POR sequencer descriptor format. The POR sequencer descriptor is comprised of the register information space **401**, the DMA descriptor space **403**, and an optional POR sequencer descriptor checksum **407**. The register information space consists of controller core register information **402** used by the POR sequencer to update the configuration registers of the system. The DMA descriptor space **403** on the other hand consists of the POR sequencer DMA descriptors' control information **404** and one or more DMA controller descriptors **406**. The DMA controller descriptors **406** are loaded by the POR Sequencer to the DMA controller's configuration register space. The POR sequencer DMA descriptors' control information **404** contains information about the DMA controller descriptors **406** of the current POR sequencer descriptor.

When the POR sequencer descriptor doesn't fit in the nonvolatile memory of the system, the descriptor is divided into fragments and distributed across the nonvolatile memory and storage devices attached to the system. FIG. **5** gives us an illustration of a fragmented POR sequencer descriptor. POR sequencer descriptor fragment 0 **501** is stored in the nonvolatile memory. Fragment 0 comprises of register information space **505**, DMA descriptor space **503**, and POR sequencer descriptor checksum **504**. The DMA descriptor space **503** is composed of DMA controller descriptors 0 to P. The last DMA controller descriptor **506** of the POR sequencer descriptor fragment **501**, **502** points to the next POR sequencer descriptor fragment **502**. POR sequencer fragment 1 to N **502** have only two parts, the register information space **505** and DMA descriptor space **503** in contrast to POR sequencer descriptor fragment 0 **501** which is made up of three parts. The POR sequencer descriptor fragment **501** or **502** may or may not have a register information space **505**. The last DMA controller descriptor **507** of the last POR sequencer descriptor fragment points to the last fragment of the system firmware in the storage device.

FIG. **6** is a flowchart showing the process of transferring the POR sequencer descriptors **212**, **213** from nonvolatile memory **215** and storage device **209** to the RAM **208**. After the system is powered-up and the input power supply stabilizes in block **601**, the CPU bus **202**, reset controller **206**, DMA controller **203**, storage device controller **204**, RAM controller **205**, nonvolatile memory controller **214** and POR sequencer **207** are brought out of reset in block **602**. The POR sequencer **207** then detects the presence of POR sequencer descriptor **212** in the nonvolatile memory in block **603**. If the first copy of the POR Sequencer descriptor **212** is detected, then POR sequencer **207** copies the POR

sequencer descriptor **212** from the nonvolatile memory **215** to the RAM **208** in block **606**. POR sequencer **207** then checks the integrity of POR sequencer descriptor **212** in block **606** by comparing the computed checksum to the POR sequencer descriptor checksum **507**. If no error is found then copying of POR sequencer descriptor **212** from nonvolatile memory **215** to the RAM **208** is completed. If an error is found, a POR sequencer error handler is activated in block **612**. If the first copy **212** is unsuccessfully detected in step **604**, in block **609**, the POR sequencer will try to detect a second copy of the POR sequencer descriptor which is not shown in FIG. **2**. If successful, the POR sequencer **207** will copy the backup copy of the POR sequencer descriptor from the nonvolatile memory **215** to the RAM **208** in block **605**. If not, the POR sequencer **207** releases the debug interface **216** and a CPU **201** from reset to allow a system integrator from updating the nonvolatile memory **215** with a new POR sequencer descriptor **212**.

FIG. **7** illustrates the register initialization phase by the POR sequencer **207**. Starting from block **701**, the POR sequencer **207** reads the first register information in the register information space **401** of the POR sequencer descriptor **212** from the RAM **208**. The POR sequencer **207** then processes the register information and then updates the corresponding configuration register based on the contents of the register information in block **702**. Steps **701** and **702** are repeated for succeeding register information until the last register information of the current POR sequencer descriptor **212** is read and processed. After the last register information is read, the POR sequencer **207** writes the DMA controller descriptor **406** to the DMA controller's register space in block **704**. The POR sequencer **207** then activates the DMA controller **203** in block **705**. The DMA controller **203** then processes the DMA controller's descriptor **406** that it received and initiates transfer of data from the storage device **209** to the RAM **208** via the storage device controller **204** in block **706**. When the data transfer is finished, in block **707**, the DMA controller **203** informs the POR sequencer **207** that the data transfer has completed. The POR sequencer **207** then reads the POR sequencer DMA descriptor's control information **404** in block **608**. It then checks if the DMA controller descriptor **406** points to a new set of register information of another POR sequencer descriptor fragment **502** in block **709**. If the test is positive, the register initialization phase is repeated starting with block **701**. If the test is negative, then the register initialization phase has ended.

DMA controller descriptors **406** are not only used to fetch POR sequencer descriptor fragments **501**, **502**. They are also used to transfer system firmware **211** from the storage device to the RAM **208**. In FIG. **8**, the copying of system firmware **211** from the storage device **209** to the RAM **208** is illustrated. The process starts with the POR sequencer **207** reading a DMA controller descriptor **406** from RAM **208** and then writing the DMA controller descriptor **406** to the configuration space of an available DMA controller **203** in block **801**. The POR sequencer **207** then activates the DMA controller **203** in block **802**, which would then cause the DMA controller **203** to start the transfer of a system firmware fragment **211** from the storage device **209** to the internal RAM **208** via the storage device controller **204** in block **803**.

The POR sequencer **207** then reads the DMA descriptor control information **404** from RAM **208** in block **804**. If the DMA descriptor **406** points to the last system firmware fragment, the POR sequencer **207** waits for all busy DMA controllers to complete their transfers in block **809**. The POR sequencer **207** then verifies the integrity of the copied system firmware in block **811**. If the system firmware is error free then the sequence completes. If an error is found, an error handling routine will be executed in block **813**. In block **805**, if the DMA controller descriptor **406** does not point to the last system firmware fragment, which means there are succeeding system firmware fragments, then the POR sequencer **207** proceeds to block **806** to check if the said DMA controller descriptor **406** was the last descriptor of the current POR sequencer descriptor. If it is the last DMA controller descriptor **406** of the current frame, then POR sequencer **207** waits for all the busy DMA controllers to complete their transfers in block **810**. The process is repeated starting from block **801** for the next POR sequencer descriptor which will be used to fetch the next system firmware fragment **211**.

In block **806**, if the DMA controller descriptor **406** of the current POR sequencer descriptor is not the last one, then the availability of a DMA controller **203** is determined in block **807**. If there is no available DMA controller, meaning, all the DMA controllers **203** are busy, then the POR sequencer **207** waits for a DMA controller **203** to complete its transfer in block **808** before proceeding to process the next DMA controller descriptor **406**. When a DMA controller becomes available the process goes back to block **801**.

FIG. **9** describes the process of releasing the CPUs **201** from their reset state. The POR sequencer **207** first configures the reset controller **206** to release a CPU **201** from reset in block **901**. This CPU is usually referred to as the bootstrap CPU. After coming out of reset, the bootstrap CPU reads and executes the system firmware from RAM **208** in block **902**. Then the bootstrap CPU performs full system initialization and diagnostics in block **903**. If the diagnostic failed, a system firmware error handling routine is executed in block **905**. If the system diagnostics is successful, the release of other CPUs, if present, proceeds. The system firmware instructs the reset controller **206** to release the next CPU from reset in block **908**. The next CPU is released from reset which then reads and executes a copy of the system firmware in block **909**. This process ends when all of the CPUs **203** have been brought out of reset. At this point, the system firmware exits system initialization phase and waits for service calls from the controller cores of the embedded system.

Foregoing described embodiments of the invention are provided as illustrations and descriptions. They are not intended to limit the invention to precise form described. In particular, it is contemplated that functional implementation of invention described herein may be implemented equivalently in hardware, software, firmware, and/or other available functional components or building blocks, and that networks may be wired, wireless, or a combination of wired and wireless.

It is also within the scope of the present invention to implement a program or code that can be stored in a machine-readable or computer-readable medium to permit a computer to perform any of the inventive techniques described above, or a program or code that can be stored in an article of manufacture that includes a computer readable medium on which computer-readable instructions for carrying out embodiments of the inventive techniques are stored. Other variations and modifications of the above-described embodiments and methods are possible in light of the teaching discussed herein.

The above description of illustrated embodiments of the invention, including what is described in the Abstract, is not intended to be exhaustive or to limit the invention to the precise forms disclosed. While specific embodiments of, and

examples for, the invention are described herein for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize.

These modifications can be made to the invention in light of the above detailed description. The terms used in the following claims should not be construed to limit the invention to the specific embodiments disclosed in the specification and the claims. Rather, the scope of the invention is to be determined entirely by the following claims, which are to be construed in accordance with established doctrines of claim interpretation.

What is claimed is:

1. A computer-implemented method, comprising:

releasing components of an embedded system from reset;

detecting a power-on reset (POR) sequencer descriptor in a nonvolatile memory;

wherein the POR sequencer descriptor comprises information to initialize configuration registers of the embedded system, Direct Memory Access (DMA) descriptors used to fetch other POR sequencer descriptor fragments, and a system firmware;

copying the POR sequencer descriptor from the nonvolatile memory to a random access memory (RAM);

verifying an integrity of the POR sequencer descriptor; and

detecting any error in the POR sequencer descriptor;

wherein the computer-implemented method allows the embedded system to verify an integrity of user data and program code used in a boot process in order to provide a resilient boot up sequence and wherein the computer-implemented method reduces an amount of the nonvolatile memory used in the boot process.

2. The method of claim **1**, further comprising:

if a copy of the POR sequencer descriptor is not detected in the nonvolatile memory, then detecting a second copy of the POR sequencer descriptor and copying the second copy from the nonvolatile memory to the RAM.

3. The method of claim **1**, further comprising:

if a copy of the POR sequencer descriptor is not successfully detected, then updating the nonvolatile memory with a new POR sequencer descriptor.

4. The method of claim **1**, wherein verifying the integrity comprises comparing a computed checksum to a checksum of the POR sequencer descriptor.

5. The method of claim **1**, further comprising

activating a POR sequencer error handler if an error is detected in the POR sequencer descriptor.

6. The method of claim **1**, wherein the components comprises a processor bus, a reset controller, a direct memory access (DMA) controller, a storage device controller, a RAM controller, nonvolatile memory controller, and a POR sequencer.

7. The method of claim **1**, further comprising: stabilizing an input power supply to the embedded system after power-up.

8. The method of claim **1**, further comprising:

performing a register initialization phase in the embedded system.

9. The method of claim **1**, further comprising:

copying system firmware from a storage device to the RAM.

10. An apparatus, comprising:

an embedded system configured to release components of the embedded system from reset;

wherein the embedded system comprises a power-on reset (POR) sequencer configured to detect a (POR) sequencer descriptor in a nonvolatile memory, to copy the POR sequencer descriptor from the nonvolatile memory to a random access memory (RAM), to verify an integrity of the POR sequencer descriptor, and to detect any error in the POR sequencer descriptor;

wherein the POR sequencer descriptor comprises information to initialize configuration registers of the embedded system, Direct Memory Access (DMA) descriptors used to fetch other POR sequencer descriptor fragments, and a system firmware;

wherein the POR sequencer comprises a bus master interface that manages data and control signals to and from a Central Processor Unit (CPU) bus, a memory controller port interface that manages control signals to and from the RAM, and a POR sequencer state machine that utilizes the bus master interface to perform data transfer to and from the CPU bus and utilizes the memory controller port interface to perform data transfer to and from the RAM.

11. The apparatus of claim **10**, wherein if a copy of the POR sequencer descriptor is not detected in the nonvolatile memory, then the POR sequencer is configured to detect a second copy of the POR sequencer descriptor and to copy the second copy from the nonvolatile memory to the RAM.

12. The apparatus of claim **10**, wherein if a copy of the POR sequencer descriptor is not successfully detected, then a system integrator is configured to update the nonvolatile memory with a new POR sequencer descriptor.

13. The apparatus of claim **10**, wherein the POR sequencer is configured to verify the integrity by comparing a computed checksum to a checksum of the POR sequencer descriptor.

14. The apparatus of claim **10**, wherein a POR sequencer error handler is activated if an error is detected in the POR sequencer descriptor.

15. The apparatus of claim **10**, wherein the components comprises a processor bus, a reset controller, a direct memory access (DMA) controller, a storage device controller, a RAM controller, nonvolatile memory controller, and a POR sequencer.

16. The apparatus of claim **10**, wherein an input power supply to the embedded system is stabilized after power-up.

17. The apparatus of claim **10**, wherein the POR sequencer is configured to perform a register initialization phase in the embedded system.

18. The apparatus of claim **10**, wherein the embedded system is configured to copy system firmware from a storage device to the RAM.

19. An article of manufacture, comprising:

a non-transitory computer-readable medium having stored thereon instructions operable to permit an apparatus to perform a method comprising:

releasing components of an embedded system from reset;

detecting a power-on reset (POR) sequencer descriptor in a nonvolatile memory;

wherein the POR sequencer descriptor comprises information to initialize configuration registers of the embedded system, Direct Memory Access (DMA) descriptors used to fetch other POR sequencer descriptor fragments, and a system firmware;

copying the POR sequencer descriptor from the nonvolatile memory to a random access memory;

verifying an integrity of the POR sequencer descriptor; and

detecting any error in the POR sequencer descriptor.

20. The article of manufacture of claim **19**, wherein the instructions are further operable to permit the apparatus to

activate a POR sequencer error handler if an error is detected in the POR sequencer descriptor.

* * * * *