US Patent & Trademark Office
Patent Public Search | Text View

United States Patent 12395429
Kind Code B2
Date of Patent August 19, 2025
Inventor(s) Jha; Prakash Kumar Arvind et al.

Medical device communication method

Abstract

A medical device communication method that may be implemented within a variety of medical devices including but not limited to infusion pumps. The method may be implemented with a protocol stack for at least intra-device communication. Embodiments provide connection-oriented, connectionless-oriented, broadcast and multicast data exchange with priority handling of data, fragmentation, and reassembly of data, unique static and dynamic address assignment and hot swap capability for connected peripherals or subsystems.

Inventors: **Jha; Prakash Kumar Arvind (San Diego, CA), Cudney; James (Santee, CA), Herr; Benjamin (Kelso, WA), Lee; Mark I. (Poway, CA), Picinich; Matteo D. (Temecula, CA)**

Applicant: **ICU Medical, Inc.** (San Clemente, CA)

Family ID: **1000008766954**

Assignee: **ICU Medical, Inc. (San Clemente, CA)**

Appl. No.: **18/743658**

Filed: **June 14, 2024**

Prior Publication Data

Document Identifier	Publication Date
US 20250016093 A1	Jan. 09, 2025

Related U.S. Application Data

continuation parent-doc US 18045061 20221007 US 12047292 child-doc US 18743658
continuation parent-doc US 17074268 20201019 US 11470000 20221011 child-doc US 18045061
continuation parent-doc US 16446381 20190619 US 10812380 20201020 child-doc US 17074268
continuation parent-doc US 15581442 20170428 US 10333843 20190625 child-doc US 16446381
continuation parent-doc US 14198807 20140306 US 9641432 20170502 child-doc US 15581442
us-provisional-application US 61773647 20130306

Publication Classification

Int. Cl.: **H04L45/74** (20220101); **G16H20/17** (20180101); **G16H40/63** (20180101); **G16H40/67** (20180101); **H04L45/302** (20220101); **H04L69/16** (20220101); **H04L69/165** (20220101); **H04L69/32** (20220101)

U.S. Cl.:

CPC **H04L45/74** (20130101); **G16H20/17** (20180101); **G16H40/63** (20180101); **G16H40/67** (20180101); **H04L45/302** (20130101); **H04L69/16** (20130101); **H04L69/162** (20130101); **H04L69/165** (20130101); **H04L69/32** (20130101);

Field of Classification Search

CPC: H04L (69/32); H04L (69/165); H04L (69/162); H04L (69/16); H04L (45/74); H04L (45/302); G16H (40/67); G16H (40/63); G16H (20/17)

USPC: 370/392

References Cited

U.S. PATENT DOCUMENTS

Patent No.	Issued Date	Patentee Name	U.S. Cl.	CPC
4024864	12/1976	Davies et al.	N/A	N/A
4055175	12/1976	Clemens et al.	N/A	N/A
4151845	12/1978	Clemens	N/A	N/A
4213454	12/1979	Shim	N/A	N/A
4240438	12/1979	Updike et al.	N/A	N/A

4280494	12/1980	Cosgrove et al.	N/A	N/A
4308866	12/1981	Jeliffe	N/A	N/A
4370983	12/1982	Lichtenstein et al.	N/A	N/A
4373527	12/1982	Fischell	N/A	N/A
4392849	12/1982	Petre et al.	N/A	N/A
4395259	12/1982	Prestele et al.	N/A	N/A
4457751	12/1983	Rodler	N/A	N/A
4464170	12/1983	Clemens	N/A	N/A
4469481	12/1983	Kobayashi	N/A	N/A
4475901	12/1983	Kraegen et al.	N/A	N/A
4494950	12/1984	Fischell	N/A	N/A
4498843	12/1984	Schneider et al.	N/A	N/A
4515584	12/1984	Abe et al.	N/A	N/A
4526568	12/1984	Clemens et al.	N/A	N/A
4529401	12/1984	Leslie et al.	N/A	N/A
4543955	12/1984	Schroeppel	N/A	N/A
4551133	12/1984	Zegers de Beyl et al.	N/A	N/A
4553958	12/1984	LeCocq	N/A	N/A
4559037	12/1984	Franetzki et al.	N/A	N/A
4613937	12/1985	Batty	N/A	N/A
4624661	12/1985	Arimond	N/A	N/A
4633878	12/1986	Bombardieri	N/A	N/A
4634426	12/1986	Kamen	N/A	N/A
4634427	12/1986	Hannula et al.	N/A	N/A
4674652	12/1986	Aten et al.	N/A	N/A
4676776	12/1986	Howson et al.	N/A	N/A
4679562	12/1986	Luksha	N/A	N/A
4685903	12/1986	Cable et al.	N/A	N/A
4695954	12/1986	Rose	N/A	N/A
4696671	12/1986	Epstein et al.	N/A	N/A
4714462	12/1986	DiDomenico	N/A	N/A
4722734	12/1987	Kolin	N/A	N/A
4730849	12/1987	Siegel	N/A	N/A
4731051	12/1987	Fischell	N/A	N/A
4741732	12/1987	Crankshaw et al.	N/A	N/A
4756706	12/1987	Kerns et al.	N/A	N/A
4776842	12/1987	Franetzki et al.	N/A	N/A
4785969	12/1987	McLaughlin	N/A	N/A
4803625	12/1988	Fu et al.	N/A	N/A
4835372	12/1988	Gombrich et al.	N/A	N/A
4838275	12/1988	Lee	N/A	N/A
4838856	12/1988	Mulreany et al.	N/A	N/A
4838857	12/1988	Strowe et al.	N/A	N/A
4854324	12/1988	Hirschman et al.	N/A	N/A
4857716	12/1988	Gombrich et al.	N/A	N/A
4858154	12/1988	Anderson et al.	N/A	N/A
4898578	12/1989	Rubalcaba, Jr.	N/A	N/A
4908017	12/1989	Howson et al.	N/A	N/A
4933873	12/1989	Kaufman et al.	N/A	N/A
4943279	12/1989	Samiotes et al.	N/A	N/A
4946439	12/1989	Eggers	N/A	N/A
4953745	12/1989	Rowlett	N/A	N/A
4978335	12/1989	Arthur, III	N/A	N/A
5000739	12/1990	Kulisz et al.	N/A	N/A
5010473	12/1990	Jacobs	N/A	N/A
5014698	12/1990	Cohen	N/A	N/A
5016172	12/1990	Dessertine	N/A	N/A
5026084	12/1990	Paisfield	N/A	N/A
5034004	12/1990	Crankshaw	N/A	N/A
5041086	12/1990	Koenig et al.	N/A	N/A
5058161	12/1990	Weiss	N/A	N/A
5078683	12/1991	Sancoff et al.	N/A	N/A
5084828	12/1991	Kaufman et al.	N/A	N/A
5088981	12/1991	Howson et al.	N/A	N/A
5097505	12/1991	Weiss	N/A	N/A
5100380	12/1991	Epstein et al.	N/A	N/A
5102392	12/1991	Sakai et al.	N/A	N/A
5104374	12/1991	Bishko et al.	N/A	N/A
5109850	12/1991	Blanco et al.	N/A	N/A
5131816	12/1991	Brown	N/A	N/A
5142484	12/1991	Kaufman et al.	N/A	N/A
5153827	12/1991	Coutre et al.	N/A	N/A
5157640	12/1991	Backner	N/A	N/A
5161222	12/1991	Montejo et al.	N/A	N/A
5177993	12/1992	Beckman et al.	N/A	N/A

5181910	12/1992	Scanlon	N/A	N/A
5190522	12/1992	Wocicki et al.	N/A	N/A
5199439	12/1992	Zimmerman et al.	N/A	N/A
5200891	12/1992	Kehr et al.	N/A	N/A
5216597	12/1992	Beckers	N/A	N/A
5221268	12/1992	Barton et al.	N/A	N/A
5230061	12/1992	Welch	N/A	N/A
5243982	12/1992	Möstl et al.	N/A	N/A
5244463	12/1992	Cordner, Jr. et al.	N/A	N/A
5249260	12/1992	Nigawara et al.	N/A	N/A
5254096	12/1992	Rondelet et al.	N/A	N/A
5256156	12/1992	Kern et al.	N/A	N/A
5256157	12/1992	Samiotes et al.	N/A	N/A
5261702	12/1992	Mayfield	N/A	N/A
5317506	12/1993	Coutre et al.	N/A	N/A
5319355	12/1993	Russek	N/A	N/A
5319363	12/1993	Welch et al.	N/A	N/A
5330634	12/1993	Wong et al.	N/A	N/A
5338157	12/1993	Blomquist	N/A	N/A
5341476	12/1993	Lowell	N/A	N/A
5364346	12/1993	Schrezenmeir	N/A	N/A
5366346	12/1993	Danby	N/A	N/A
5368562	12/1993	Blomquist et al.	N/A	N/A
5373454	12/1993	Kanda et al.	N/A	N/A
5376070	12/1993	Purvis et al.	N/A	N/A
5378231	12/1994	Johnson et al.	N/A	N/A
5389071	12/1994	Kawahara et al.	N/A	N/A
5389078	12/1994	Zalesky et al.	N/A	N/A
5417222	12/1994	Dempsey et al.	N/A	N/A
5423748	12/1994	Uhala	N/A	N/A
5429602	12/1994	Hauser	N/A	N/A
5431627	12/1994	Pastrone et al.	N/A	N/A
5432777	12/1994	Le Boudec et al.	N/A	N/A
5445621	12/1994	Poli et al.	N/A	N/A
5447164	12/1994	Shaya et al.	N/A	N/A
5455851	12/1994	Chaco et al.	N/A	N/A
5461365	12/1994	Schlager et al.	N/A	N/A
5464392	12/1994	Epstein et al.	N/A	N/A
5465082	12/1994	Chaco	N/A	N/A
5485408	12/1995	Blomquist	N/A	N/A
5486286	12/1995	Peterson et al.	N/A	N/A
5493430	12/1995	Lu et al.	N/A	N/A
5496273	12/1995	Pastrone et al.	N/A	N/A
5505828	12/1995	Wong et al.	N/A	N/A
5507288	12/1995	Bocker et al.	N/A	N/A
5507786	12/1995	Morgan et al.	N/A	N/A
5508499	12/1995	Ferrario	N/A	N/A
5515713	12/1995	Saugues et al.	N/A	N/A
5520637	12/1995	Pager et al.	N/A	N/A
5522798	12/1995	Johnson et al.	N/A	N/A
5547470	12/1995	Johnson et al.	N/A	N/A
5554013	12/1995	Owens et al.	N/A	N/A
5562615	12/1995	Nassif	N/A	N/A
5577169	12/1995	Prezioso	N/A	N/A
5582323	12/1995	Kurtenbach	N/A	N/A
5582593	12/1995	Hultman	N/A	N/A
5594786	12/1996	Chaco et al.	N/A	N/A
5598519	12/1996	Narayanan	N/A	N/A
5620608	12/1996	Rosa et al.	N/A	N/A
5630710	12/1996	Tune et al.	N/A	N/A
5636044	12/1996	Yuan et al.	N/A	N/A
5643212	12/1996	Coutre et al.	N/A	N/A
5651775	12/1996	Walker et al.	N/A	N/A
5658131	12/1996	Aoki et al.	N/A	N/A
5658250	12/1996	Blomquist et al.	N/A	N/A
5665065	12/1996	Colman et al.	N/A	N/A
5669877	12/1996	Blomquist	N/A	N/A
5672154	12/1996	Sillén et al.	N/A	N/A
5681285	12/1996	Ford et al.	N/A	N/A
5685844	12/1996	Marttila	N/A	N/A
5687717	12/1996	Halpern et al.	N/A	N/A
5689229	12/1996	Chaco et al.	N/A	N/A
5697899	12/1996	Hillman et al.	N/A	N/A
5699509	12/1996	Gary et al.	N/A	N/A
5708714	12/1997	Lopez et al.	N/A	N/A

5713350	12/1997	Yokota et al.	N/A	N/A
5713856	12/1997	Eggers et al.	N/A	N/A
5718562	12/1997	Lawless et al.	N/A	N/A
5719761	12/1997	Gatti et al.	N/A	N/A
5733259	12/1997	Valcke et al.	N/A	N/A
5738102	12/1997	Lemelson	N/A	N/A
5744027	12/1997	Connell et al.	N/A	N/A
5752621	12/1997	Passamante	N/A	N/A
5754111	12/1997	Garcia	N/A	N/A
5764034	12/1997	Bowman et al.	N/A	N/A
5764159	12/1997	Neftel et al.	N/A	N/A
5772635	12/1997	Dastur et al.	N/A	N/A
5774865	12/1997	Glynn	N/A	N/A
5778256	12/1997	Darbee	N/A	N/A
5778345	12/1997	McCartney	N/A	N/A
5781442	12/1997	Engleson et al.	N/A	N/A
5782805	12/1997	Meinzer et al.	N/A	N/A
5788669	12/1997	Peterson	N/A	N/A
5797515	12/1997	Liff et al.	N/A	N/A
5800387	12/1997	Duffy et al.	N/A	N/A
5814015	12/1997	Gargano et al.	N/A	N/A
5822544	12/1997	Chaco et al.	N/A	N/A
5822715	12/1997	Worthington et al.	N/A	N/A
5827179	12/1997	Lichter et al.	N/A	N/A
5832448	12/1997	Brown	N/A	N/A
5836910	12/1997	Duffy et al.	N/A	N/A
5850344	12/1997	Conkright	N/A	N/A
5867821	12/1998	Ballantyne et al.	N/A	N/A
5870733	12/1998	Bass et al.	N/A	N/A
5871465	12/1998	Vasko	N/A	N/A
5873731	12/1998	Predergast	N/A	N/A
5885245	12/1998	Lynch et al.	N/A	N/A
5897493	12/1998	Brown	N/A	N/A
5897498	12/1998	Canfield, II et al.	N/A	N/A
5910252	12/1998	Truitt et al.	N/A	N/A
5912818	12/1998	McGrady et al.	N/A	N/A
5915240	12/1998	Karpf	N/A	N/A
5920054	12/1998	Uber, III	N/A	N/A
5920263	12/1998	Huttenhoff et al.	N/A	N/A
5924074	12/1998	Evans	N/A	N/A
5931764	12/1998	Freeman et al.	N/A	N/A
5935099	12/1998	Peterson et al.	N/A	N/A
5935106	12/1998	Olsen	N/A	N/A
5941846	12/1998	Duffy et al.	N/A	N/A
5956501	12/1998	Brown	N/A	N/A
5957885	12/1998	Bollish et al.	N/A	N/A
5960085	12/1998	de la Huerga	N/A	N/A
5961448	12/1998	Swenson et al.	N/A	N/A
5967559	12/1998	Abramowitz	N/A	N/A
5971594	12/1998	Sahai et al.	N/A	N/A
5975081	12/1998	Hood et al.	N/A	N/A
5990838	12/1998	Burns et al.	N/A	N/A
5997476	12/1998	Brown	N/A	N/A
6000828	12/1998	Leet	N/A	N/A
6003006	12/1998	Colella et al.	N/A	N/A
6012034	12/1999	Hamparian et al.	N/A	N/A
6017318	12/1999	Gauthier et al.	N/A	N/A
6021392	12/1999	Lester et al.	N/A	N/A
6024539	12/1999	Blomquist	N/A	N/A
6024699	12/1999	Surwit et al.	N/A	N/A
6032155	12/1999	de la Huerga	N/A	N/A
6032676	12/1999	Moore	N/A	N/A
6039251	12/1999	Holowko et al.	N/A	N/A
6070761	12/1999	Bloom et al.	N/A	N/A
6073106	12/1999	Rozen et al.	N/A	N/A
6104295	12/1999	Gaisser et al.	N/A	N/A
6112182	12/1999	Akers et al.	N/A	N/A
6112323	12/1999	Meizlik et al.	N/A	N/A
RE36871	12/1999	Epstein et al.	N/A	N/A
6115365	12/1999	Newberg	N/A	N/A
6115390	12/1999	Chuah	N/A	N/A
6122536	12/1999	Sun et al.	N/A	N/A
6126637	12/1999	Kriesel et al.	N/A	N/A
6135949	12/1999	Russo et al.	N/A	N/A
6150942	12/1999	O'Brien	N/A	N/A

6151643	12/1999	Cheng et al.	N/A	N/A
6157914	12/1999	Seto et al.	N/A	N/A
6159147	12/1999	Lichter et al.	N/A	N/A
6167567	12/1999	Chiles et al.	N/A	N/A
6182667	12/2000	Hanks et al.	N/A	N/A
6189105	12/2000	Lopes	N/A	N/A
6195589	12/2000	Ketcham	N/A	N/A
6208974	12/2000	Campbell et al.	N/A	N/A
6222323	12/2000	Yamashita et al.	N/A	N/A
6223440	12/2000	Rashman	N/A	N/A
6226277	12/2000	Chuah	N/A	N/A
6227371	12/2000	Song	N/A	N/A
6234176	12/2000	Domae et al.	N/A	N/A
6241704	12/2000	Peterson et al.	N/A	N/A
6248067	12/2000	Causey, III et al.	N/A	N/A
6249705	12/2000	Snell	N/A	N/A
6257265	12/2000	Brunner et al.	N/A	N/A
6259355	12/2000	Chaco et al.	N/A	N/A
6269340	12/2000	Ford et al.	N/A	N/A
6270455	12/2000	Brown	N/A	N/A
6271813	12/2000	Palalau	N/A	N/A
6277072	12/2000	Bardy	N/A	N/A
6280380	12/2000	Bardy	N/A	N/A
6283761	12/2000	Joao	N/A	N/A
6285665	12/2000	Chuah	N/A	N/A
6292860	12/2000	Cochcroft, Jr.	N/A	N/A
6312378	12/2000	Bardy	N/A	N/A
6327254	12/2000	Chuah	N/A	N/A
6330008	12/2000	Razdow et al.	N/A	N/A
6339718	12/2001	Zatezalo et al.	N/A	N/A
6346886	12/2001	de la Huerga	N/A	N/A
6363282	12/2001	Nichols et al.	N/A	N/A
6371719	12/2001	Hildebrandt	N/A	N/A
6377548	12/2001	Chuah	N/A	N/A
6388951	12/2001	Matsumoto et al.	N/A	N/A
6406426	12/2001	Reuss et al.	N/A	N/A
6408330	12/2001	Huerga	N/A	N/A
6418334	12/2001	Unger et al.	N/A	N/A
6427088	12/2001	Bowman et al.	N/A	N/A
6428483	12/2001	Carlebach	N/A	N/A
6442432	12/2001	Lee	N/A	N/A
6469991	12/2001	Chuah	N/A	N/A
6475180	12/2001	Peterson et al.	N/A	N/A
6482158	12/2001	Mault	N/A	N/A
6485418	12/2001	Yasushi et al.	N/A	N/A
6494694	12/2001	Lawless et al.	N/A	N/A
6494831	12/2001	Koritzinsky	N/A	N/A
6497680	12/2001	Holst et al.	N/A	N/A
6514460	12/2002	Fendrock	N/A	N/A
6517482	12/2002	Eiden et al.	N/A	N/A
6519569	12/2002	White et al.	N/A	N/A
6520930	12/2002	Critchlow et al.	N/A	N/A
6540672	12/2002	Simonsen et al.	N/A	N/A
6542902	12/2002	Dulong et al.	N/A	N/A
6544212	12/2002	Galley et al.	N/A	N/A
6544228	12/2002	Heitmeier	N/A	N/A
6546350	12/2002	Hartmann et al.	N/A	N/A
6551276	12/2002	Mann et al.	N/A	N/A
6554798	12/2002	Mann et al.	N/A	N/A
6558320	12/2002	Causey et al.	N/A	N/A
6558351	12/2002	Steil et al.	N/A	N/A
6565509	12/2002	Say et al.	N/A	N/A
6567416	12/2002	Chuah	N/A	N/A
6571294	12/2002	Simmon et al.	N/A	N/A
6572542	12/2002	Houben et al.	N/A	N/A
6572545	12/2002	Knobbe et al.	N/A	N/A
6578002	12/2002	Derzay et al.	N/A	N/A
6581117	12/2002	Klein et al.	N/A	N/A
6587034	12/2002	Heiman et al.	N/A	N/A
6589229	12/2002	Connelly et al.	N/A	N/A
6599281	12/2002	Struys et al.	N/A	N/A
6602191	12/2002	Quy	N/A	N/A
6605072	12/2002	Struys et al.	N/A	N/A
6628809	12/2002	Rowe et al.	N/A	N/A
6631353	12/2002	Davis et al.	N/A	N/A

6640246	12/2002	Gardy, Jr. et al.	N/A	N/A
6641533	12/2002	Causey, III et al.	N/A	N/A
6647299	12/2002	Bourget	N/A	N/A
6652455	12/2002	Kocher	N/A	N/A
6653937	12/2002	Nelson et al.	N/A	N/A
6659947	12/2002	Carter et al.	N/A	N/A
6669630	12/2002	Joliat et al.	N/A	N/A
6671563	12/2002	Engleson et al.	N/A	N/A
6673033	12/2003	Sciulli et al.	N/A	N/A
6674403	12/2003	Gray et al.	N/A	N/A
6681003	12/2003	Linder et al.	N/A	N/A
6689091	12/2003	Bui et al.	N/A	N/A
6692241	12/2003	Watanabe et al.	N/A	N/A
6694191	12/2003	Starkweather et al.	N/A	N/A
6694334	12/2003	DuLong et al.	N/A	N/A
6721286	12/2003	Williams et al.	N/A	N/A
6721582	12/2003	Trepagnier et al.	N/A	N/A
6725200	12/2003	Rost	N/A	N/A
6731989	12/2003	Engleson et al.	N/A	N/A
6740072	12/2003	Starkweather et al.	N/A	N/A
6751651	12/2003	Crockett	N/A	N/A
6752787	12/2003	Causey, III et al.	N/A	N/A
6753830	12/2003	Gelbman	N/A	N/A
6758810	12/2003	Lebel et al.	N/A	N/A
6773396	12/2003	Flach et al.	N/A	N/A
6774786	12/2003	Havekost et al.	N/A	N/A
6775577	12/2003	Cmkovich et al.	N/A	N/A
6780156	12/2003	Haueter et al.	N/A	N/A
6790198	12/2003	White et al.	N/A	N/A
6792470	12/2003	Hakenberg et al.	N/A	N/A
6796956	12/2003	Hartlaub et al.	N/A	N/A
6799149	12/2003	Hartlaub	N/A	N/A
6809653	12/2003	Mann et al.	N/A	N/A
6811534	12/2003	Bowman, IV et al.	N/A	N/A
6816605	12/2003	Rowe et al.	N/A	N/A
6839753	12/2004	Biondi et al.	N/A	N/A
6852104	12/2004	Blomquist	N/A	N/A
6859134	12/2004	Heiman et al.	N/A	N/A
6871211	12/2004	Labounty et al.	N/A	N/A
6873268	12/2004	Lebel et al.	N/A	N/A
6876303	12/2004	Reeder et al.	N/A	N/A
6885881	12/2004	Leonhardt	N/A	N/A
6891525	12/2004	Ogoro	N/A	N/A
6892278	12/2004	Ebergen	N/A	N/A
6899695	12/2004	Herrera	N/A	N/A
6915170	12/2004	Engleson et al.	N/A	N/A
6923763	12/2004	Kovatchev et al.	N/A	N/A
6924781	12/2004	Gelbman	N/A	N/A
6928338	12/2004	Buchser et al.	N/A	N/A
6928490	12/2004	Bucholz et al.	N/A	N/A
6936029	12/2004	Mann et al.	N/A	N/A
6945954	12/2004	Hochman et al.	N/A	N/A
6948492	12/2004	Wemeling et al.	N/A	N/A
6958677	12/2004	Carter	N/A	N/A
6958691	12/2004	Anderson et al.	N/A	N/A
6958705	12/2004	Lebel et al.	N/A	N/A
6961448	12/2004	Nichols et al.	N/A	N/A
6969352	12/2004	Chiang et al.	N/A	N/A
6969865	12/2004	Duchon et al.	N/A	N/A
6974437	12/2004	Lebel et al.	N/A	N/A
6979326	12/2004	Mann et al.	N/A	N/A
6980958	12/2004	Surwit et al.	N/A	N/A
6985870	12/2005	Martucci et al.	N/A	N/A
6986347	12/2005	Hickle	N/A	N/A
6997880	12/2005	Carlebach et al.	N/A	N/A
6997920	12/2005	Mann et al.	N/A	N/A
6998984	12/2005	Zittrain	N/A	N/A
7016752	12/2005	Ruben et al.	N/A	N/A
7017293	12/2005	Riley	N/A	N/A
7025743	12/2005	Mann et al.	N/A	N/A
7029455	12/2005	Flaherty	N/A	N/A
7038584	12/2005	Carter	N/A	N/A
7060031	12/2005	Webb et al.	N/A	N/A
7060059	12/2005	Keith et al.	N/A	N/A
7069552	12/2005	Lindberg et al.	N/A	N/A

7072725	12/2005	Bristol et al.	N/A	N/A
7079035	12/2005	Bock et al.	N/A	N/A
7092943	12/2005	Roese et al.	N/A	N/A
7096072	12/2005	Engleson et al.	N/A	N/A
7099809	12/2005	Dori	N/A	N/A
7103419	12/2005	Engleson et al.	N/A	N/A
7103578	12/2005	Beck et al.	N/A	N/A
7107106	12/2005	Engleson et al.	N/A	N/A
7108680	12/2005	Rohr et al.	N/A	N/A
7109878	12/2005	Mann et al.	N/A	N/A
7114002	12/2005	Okumura et al.	N/A	N/A
7117041	12/2005	Engleson et al.	N/A	N/A
7136645	12/2005	Hanson et al.	N/A	N/A
7137964	12/2005	Flaherty	N/A	N/A
7142190	12/2005	Martinez	N/A	N/A
7150741	12/2005	Erickson et al.	N/A	N/A
7153289	12/2005	Vasko	N/A	N/A
7154397	12/2005	Zerhusen et al.	N/A	N/A
7156807	12/2006	Carter et al.	N/A	N/A
7158030	12/2006	Chung	N/A	N/A
7161484	12/2006	Tsoukalis et al.	N/A	N/A
7167755	12/2006	Seeberger et al.	N/A	N/A
7167920	12/2006	Traversat	N/A	N/A
7171277	12/2006	Engleson et al.	N/A	N/A
7171492	12/2006	Borella et al.	N/A	N/A
7181493	12/2006	English et al.	N/A	N/A
7185288	12/2006	McKeever	N/A	N/A
7193514	12/2006	Ritson	N/A	N/A
7197025	12/2006	Chuah	N/A	N/A
7201734	12/2006	Hickle	N/A	N/A
7204823	12/2006	Estes et al.	N/A	N/A
7213009	12/2006	Pestotnik	N/A	N/A
7216802	12/2006	de la Huerga	N/A	N/A
7220240	12/2006	Struys et al.	N/A	N/A
7224979	12/2006	Singhal et al.	N/A	N/A
7229430	12/2006	Hickle et al.	N/A	N/A
7230529	12/2006	Ketcherside	N/A	N/A
7236936	12/2006	White et al.	N/A	N/A
7238164	12/2006	Childers et al.	N/A	N/A
7247154	12/2006	Hickle	N/A	N/A
7248239	12/2006	Dowling	N/A	N/A
7250856	12/2006	Havekost et al.	N/A	N/A
7255683	12/2006	Vanderveen et al.	N/A	N/A
7256888	12/2006	Staehr et al.	N/A	N/A
7258534	12/2006	Fathallah et al.	N/A	N/A
7263213	12/2006	Rowe	N/A	N/A
7267664	12/2006	Rizzo	N/A	N/A
7267665	12/2006	Steil et al.	N/A	N/A
7275156	12/2006	Balfanz et al.	N/A	N/A
7278983	12/2006	Ireland et al.	N/A	N/A
7289815	12/2006	Gfeller et al.	N/A	N/A
7289948	12/2006	Mohri	N/A	N/A
7293107	12/2006	Hanson et al.	N/A	N/A
7295119	12/2006	Rappaport et al.	N/A	N/A
7295556	12/2006	Roese et al.	N/A	N/A
7301451	12/2006	Hastings	N/A	N/A
7308300	12/2006	Toews et al.	N/A	N/A
7315825	12/2007	Rosenfeld et al.	N/A	N/A
7319386	12/2007	Collins, Jr. et al.	N/A	N/A
7324000	12/2007	Zittrain et al.	N/A	N/A
7327705	12/2007	Fletcher et al.	N/A	N/A
7343224	12/2007	DiGianfilippo et al.	N/A	N/A
7346025	12/2007	Bryson	N/A	N/A
7347836	12/2007	Peterson et al.	N/A	N/A
7354420	12/2007	Steil et al.	N/A	N/A
7369897	12/2007	Boveja et al.	N/A	N/A
7369948	12/2007	Ferenczi et al.	N/A	N/A
7383088	12/2007	Spinelli et al.	N/A	N/A
7384410	12/2007	Eggers et al.	N/A	N/A
7398183	12/2007	Holland et al.	N/A	N/A
7398279	12/2007	Muno, Jr. et al.	N/A	N/A
7399277	12/2007	Saidara et al.	N/A	N/A
7402153	12/2007	Steil et al.	N/A	N/A
7420472	12/2007	Tran	N/A	N/A
7432807	12/2007	Schmitt	N/A	N/A

7436454	12/2007	Yamaguchi et al.	N/A	N/A
7447643	12/2007	Olson	N/A	N/A
7454314	12/2007	Holland et al.	N/A	N/A
7457804	12/2007	Uber, III et al.	N/A	N/A
7464040	12/2007	Joao	N/A	N/A
7469213	12/2007	Rao	N/A	N/A
7471994	12/2007	Ford et al.	N/A	N/A
7483756	12/2008	Engleson et al.	N/A	N/A
7489808	12/2008	Gerder	N/A	N/A
7490021	12/2008	Holland et al.	N/A	N/A
7490048	12/2008	Joao	N/A	N/A
7491187	12/2008	Van Den Berghe et al.	N/A	N/A
7519905	12/2008	Kougiouris et al.	N/A	N/A
7523401	12/2008	Aldridge	N/A	N/A
7524304	12/2008	Genosar	N/A	N/A
7551078	12/2008	Carlson	N/A	N/A
7559321	12/2008	Wermeling et al.	N/A	N/A
7565197	12/2008	Haulbrich et al.	N/A	N/A
7572230	12/2008	Neumann et al.	N/A	N/A
7578802	12/2008	Hickle	N/A	N/A
7621009	12/2008	Elhabashy	N/A	N/A
D606533	12/2008	De Jong et al.	N/A	N/A
7636718	12/2008	Steen et al.	N/A	N/A
7640172	12/2008	Kuth	N/A	N/A
7645258	12/2009	White et al.	N/A	N/A
7647237	12/2009	Malave et al.	N/A	N/A
7662124	12/2009	Duchon et al.	N/A	N/A
7668731	12/2009	Martucci et al.	N/A	N/A
7671733	12/2009	McNeal et al.	N/A	N/A
7678071	12/2009	Lebel et al.	N/A	N/A
7687678	12/2009	Jacobs	N/A	N/A
7697994	12/2009	VanDanacker et al.	N/A	N/A
7698239	12/2009	Lieuallen	N/A	N/A
7705727	12/2009	Pestotnik	N/A	N/A
7724147	12/2009	Brown et al.	N/A	N/A
7739126	12/2009	Cave	N/A	N/A
7746218	12/2009	Collins, Jr.	N/A	N/A
7766873	12/2009	Moberg et al.	N/A	N/A
7776029	12/2009	Whitehurst et al.	N/A	N/A
7776031	12/2009	Hartlaub et al.	N/A	N/A
7785313	12/2009	Mastrototaro	N/A	N/A
7788369	12/2009	McAllen et al.	N/A	N/A
7806852	12/2009	Jurson	N/A	N/A
7806886	12/2009	Kanderian, Jr. et al.	N/A	N/A
7826981	12/2009	Goode, Jr. et al.	N/A	N/A
7835927	12/2009	Schlotterbeck et al.	N/A	N/A
7836314	12/2009	Chieu	N/A	N/A
7856276	12/2009	Ripart et al.	N/A	N/A
7860583	12/2009	Condurso et al.	N/A	N/A
7864771	12/2010	Tavares et al.	N/A	N/A
7868754	12/2010	Salvat, Jr.	N/A	N/A
7871394	12/2010	Halbert et al.	N/A	N/A
7886231	12/2010	Hopermann et al.	N/A	N/A
7895053	12/2010	Holland et al.	N/A	N/A
7896842	12/2010	Palmroos et al.	N/A	N/A
7899546	12/2010	Sieracki et al.	N/A	N/A
7905710	12/2010	Wang et al.	N/A	N/A
7920061	12/2010	Klein et al.	N/A	N/A
7933780	12/2010	de la Huerga	N/A	N/A
7938796	12/2010	Moubayed	N/A	N/A
7945452	12/2010	Fathallah et al.	N/A	N/A
7974714	12/2010	Hoffberg	N/A	N/A
7976508	12/2010	Hoag	N/A	N/A
7996241	12/2010	Zak	N/A	N/A
8034026	12/2010	Grant	N/A	N/A
8038593	12/2010	Friedman et al.	N/A	N/A
8048040	12/2010	Kiani	N/A	N/A
8060576	12/2010	Chan et al.	N/A	N/A
8065161	12/2010	Howard et al.	N/A	N/A
8066672	12/2010	Mandro	N/A	N/A
8075514	12/2010	Butterfield et al.	N/A	N/A
8078983	12/2010	Davis et al.	N/A	N/A
8082018	12/2010	Duchon et al.	N/A	N/A
8082312	12/2010	Chan et al.	N/A	N/A
8095692	12/2011	Mehta et al.	N/A	N/A

8126730	12/2011	Dicks et al.	N/A	N/A
8147448	12/2011	Sundar et al.	N/A	N/A
8149131	12/2011	Blomquist	N/A	N/A
8169914	12/2011	Bajpai	N/A	N/A
8171094	12/2011	Chan et al.	N/A	N/A
8172798	12/2011	Hungerford et al.	N/A	N/A
8185322	12/2011	Schroeder et al.	N/A	N/A
8195478	12/2011	Petersen et al.	N/A	N/A
8206350	12/2011	Mann et al.	N/A	N/A
8219413	12/2011	Martinez et al.	N/A	N/A
8231578	12/2011	Fathallah et al.	N/A	N/A
8234128	12/2011	Martucci et al.	N/A	N/A
8267892	12/2011	Spencer et al.	N/A	N/A
8271106	12/2011	Wehba et al.	N/A	N/A
8287495	12/2011	Michaud et al.	N/A	N/A
8291337	12/2011	Gannin et al.	N/A	N/A
8298184	12/2011	DiPerna et al.	N/A	N/A
8312272	12/2011	Serenyl et al.	N/A	N/A
8352290	12/2012	Bartz et al.	N/A	N/A
8359338	12/2012	Butterfield et al.	N/A	N/A
8380536	12/2012	Howard et al.	N/A	N/A
8387112	12/2012	Ranjan et al.	N/A	N/A
8394077	12/2012	Jacobson et al.	N/A	N/A
8398592	12/2012	Leibner-Druska	N/A	N/A
8403908	12/2012	Jacobson et al.	N/A	N/A
8435206	12/2012	Evans et al.	N/A	N/A
8449523	12/2012	Brukalo et al.	N/A	N/A
8452953	12/2012	Buck et al.	N/A	N/A
8453645	12/2012	Figueiredo et al.	N/A	N/A
8472630	12/2012	Konrad et al.	N/A	N/A
8480648	12/2012	Burnett et al.	N/A	N/A
8486019	12/2012	White et al.	N/A	N/A
8489427	12/2012	Simpson et al.	N/A	N/A
8494879	12/2012	Davis et al.	N/A	N/A
8504179	12/2012	Blomquist	N/A	N/A
8517990	12/2012	Teel et al.	N/A	N/A
8518021	12/2012	Stewart et al.	N/A	N/A
8543416	12/2012	Palmroos et al.	N/A	N/A
8551038	12/2012	Tsoukalis et al.	N/A	N/A
8560345	12/2012	Wehba et al.	N/A	N/A
8567681	12/2012	Borges et al.	N/A	N/A
8577692	12/2012	Silkaitis et al.	N/A	N/A
8579884	12/2012	Lanier et al.	N/A	N/A
8626530	12/2013	Tran et al.	N/A	N/A
8655676	12/2013	Wehba et al.	N/A	N/A
8660860	12/2013	Wehba et al.	N/A	N/A
8662388	12/2013	Belkin	N/A	N/A
8666769	12/2013	Butler et al.	N/A	N/A
8667293	12/2013	Birtwhistle et al.	N/A	N/A
8687811	12/2013	Nierzwick et al.	N/A	N/A
8700421	12/2013	Feng et al.	N/A	N/A
8731960	12/2013	Butler et al.	N/A	N/A
8768719	12/2013	Wehba et al.	N/A	N/A
8771251	12/2013	Ruchti et al.	N/A	N/A
8777894	12/2013	Butterfield et al.	N/A	N/A
8777895	12/2013	Hsu et al.	N/A	N/A
8799012	12/2013	Butler et al.	N/A	N/A
8876793	12/2013	Ledford et al.	N/A	N/A
8886316	12/2013	Juels	N/A	N/A
8922330	12/2013	Moberg et al.	N/A	N/A
8936565	12/2014	Chawla	N/A	N/A
8945043	12/2014	Lee et al.	N/A	N/A
8952794	12/2014	Blomquist et al.	N/A	N/A
8959617	12/2014	Newlin et al.	N/A	N/A
8998100	12/2014	Halbert et al.	N/A	N/A
9026370	12/2014	Rubalcaba et al.	N/A	N/A
9069887	12/2014	Gupta et al.	N/A	N/A
9077544	12/2014	Baker et al.	N/A	N/A
9089642	12/2014	Murphy et al.	N/A	N/A
9114217	12/2014	Sur et al.	N/A	N/A
9123077	12/2014	Silkaitis et al.	N/A	N/A
9192712	12/2014	DeBelser et al.	N/A	N/A
9240002	12/2015	Hume et al.	N/A	N/A
9292692	12/2015	Wallrabenstein	N/A	N/A
9302035	12/2015	Marseille et al.	N/A	N/A

9313154	12/2015	Son	N/A	N/A
9381296	12/2015	Arrizza et al.	N/A	N/A
9393362	12/2015	Cozmi et al.	N/A	N/A
9430655	12/2015	Stockton et al.	N/A	N/A
9438580	12/2015	Kupper	N/A	N/A
9483615	12/2015	Roberts	N/A	N/A
9498583	12/2015	Sur et al.	N/A	N/A
9539383	12/2016	Kohlbrecher	N/A	N/A
9572923	12/2016	Howard et al.	N/A	N/A
9594875	12/2016	Arrizza et al.	N/A	N/A
9604000	12/2016	Wehba et al.	N/A	N/A
9641432	12/2016	Jha et al.	N/A	N/A
9649431	12/2016	Gray et al.	N/A	N/A
9662436	12/2016	Belkin et al.	N/A	N/A
9690909	12/2016	Stewart et al.	N/A	N/A
9707341	12/2016	Dumas, IIIet al.	N/A	N/A
9717845	12/2016	Istoc	N/A	N/A
9724470	12/2016	Day et al.	N/A	N/A
9764082	12/2016	Day et al.	N/A	N/A
9886550	12/2017	Lee et al.	N/A	N/A
9943269	12/2017	Muhsin et al.	N/A	N/A
9967739	12/2017	Proennecke et al.	N/A	N/A
9971871	12/2017	Arrizza et al.	N/A	N/A
9995611	12/2017	Ruchti et al.	N/A	N/A
10022498	12/2017	Ruchti et al.	N/A	N/A
10042986	12/2017	Ruchti et al.	N/A	N/A
10046112	12/2017	Oruklu et al.	N/A	N/A
10166328	12/2018	Oruklu et al.	N/A	N/A
10173008	12/2018	Simpson et al.	N/A	N/A
10188849	12/2018	Fangrow	N/A	N/A
10233179	12/2018	Ng et al.	N/A	N/A
10238799	12/2018	Kohlbrecher	N/A	N/A
10238801	12/2018	Wehba et al.	N/A	N/A
10242060	12/2018	Butler et al.	N/A	N/A
10300194	12/2018	Day et al.	N/A	N/A
10311972	12/2018	Kohlbrecher et al.	N/A	N/A
10314974	12/2018	Day et al.	N/A	N/A
10333843	12/2018	Jha et al.	N/A	N/A
10341866	12/2018	Spencer et al.	N/A	N/A
10409995	12/2018	Wasiq	N/A	N/A
10430761	12/2018	Hume et al.	N/A	N/A
10434246	12/2018	Silkaitis et al.	N/A	N/A
10438001	12/2018	Hariprasad	N/A	N/A
10452842	12/2018	Dhondse	N/A	N/A
10453157	12/2018	Kamen et al.	N/A	N/A
10463788	12/2018	Day	N/A	N/A
10516536	12/2018	Rommel	N/A	N/A
10617815	12/2019	Day et al.	N/A	N/A
10646651	12/2019	Day et al.	N/A	N/A
10681207	12/2019	Johnson et al.	N/A	N/A
10692595	12/2019	Xavier et al.	N/A	N/A
10728262	12/2019	Vaswani	N/A	N/A
10740436	12/2019	Moskal et al.	N/A	N/A
10741280	12/2019	Xavier et al.	N/A	N/A
10757219	12/2019	Moskal	N/A	N/A
10765799	12/2019	Belkin et al.	N/A	N/A
10799632	12/2019	Kohlbrecher	N/A	N/A
10812380	12/2019	Jha et al.	N/A	N/A
10861592	12/2019	Xavier et al.	N/A	N/A
10898641	12/2020	Day et al.	N/A	N/A
10950339	12/2020	Xavier et al.	N/A	N/A
10964428	12/2020	Xavier et al.	N/A	N/A
11013861	12/2020	Wehba et al.	N/A	N/A
11037668	12/2020	Ruchti et al.	N/A	N/A
11052193	12/2020	Day et al.	N/A	N/A
11139058	12/2020	Xavier et al.	N/A	N/A
11151290	12/2020	Karakoyunlu et al.	N/A	N/A
11152108	12/2020	Xavier et al.	N/A	N/A
11152109	12/2020	Xavier et al.	N/A	N/A
11152110	12/2020	Xavier et al.	N/A	N/A
11194810	12/2020	Butler et al.	N/A	N/A
11235100	12/2021	Howard et al.	N/A	N/A
11289183	12/2021	Kohlbrecher	N/A	N/A
11309070	12/2021	Xavier et al.	N/A	N/A
11328804	12/2021	Xavier et al.	N/A	N/A

11328805	12/2021	Xavier et al.	N/A	N/A
11373753	12/2021	Xavier et al.	N/A	N/A
11437132	12/2021	Xavier et al.	N/A	N/A
11470000	12/2021	Jha et al.	N/A	N/A
11483402	12/2021	Xavier et al.	N/A	N/A
11483403	12/2021	Xavier et al.	N/A	N/A
11501877	12/2021	Kohlbrecher et al.	N/A	N/A
11571508	12/2022	Jacobson et al.	N/A	N/A
11574721	12/2022	Kohlbrecher	N/A	N/A
11574737	12/2022	Dharwad et al.	N/A	N/A
11587669	12/2022	Xavier et al.	N/A	N/A
11590057	12/2022	Tagliamento et al.	N/A	N/A
11594326	12/2022	Xavier et al.	N/A	N/A
11605468	12/2022	Jacobson et al.	N/A	N/A
11626205	12/2022	Arrizza et al.	N/A	N/A
11628246	12/2022	Day et al.	N/A	N/A
11628254	12/2022	Day et al.	N/A	N/A
11654237	12/2022	Wehba et al.	N/A	N/A
11670416	12/2022	Xavier et al.	N/A	N/A
11763927	12/2022	Ruchti et al.	N/A	N/A
11783935	12/2022	Xavier et al.	N/A	N/A
11881297	12/2023	Xavier et al.	N/A	N/A
11923076	12/2023	Xavier et al.	N/A	N/A
11986623	12/2023	Jacobson et al.	N/A	N/A
11996188	12/2023	Arrizza et al.	N/A	N/A
12002562	12/2023	Kohlbrecher	N/A	N/A
12036390	12/2023	Wehba et al.	N/A	N/A
12040068	12/2023	Xavier et al.	N/A	N/A
12042623	12/2023	Day et al.	N/A	N/A
12042631	12/2023	Day et al.	N/A	N/A
12046361	12/2023	Xavier et al.	N/A	N/A
12047292	12/2023	Jha et al.	N/A	N/A
12097351	12/2023	Belkin et al.	N/A	N/A
2001/0016056	12/2000	Westphal et al.	N/A	N/A
2001/0029178	12/2000	Criss et al.	N/A	N/A
2001/0031944	12/2000	Peterson et al.	N/A	N/A
2001/0032099	12/2000	Joao	N/A	N/A
2001/0037060	12/2000	Thompson et al.	N/A	N/A
2001/0044731	12/2000	Coffman et al.	N/A	N/A
2001/0048027	12/2000	Walsh	N/A	N/A
2001/0051787	12/2000	Haller et al.	N/A	N/A
2001/0056358	12/2000	Dulong et al.	N/A	N/A
2002/0010595	12/2001	Kapp	N/A	N/A
2002/0013551	12/2001	Zaitsu et al.	N/A	N/A
2002/0013723	12/2001	Mise	N/A	N/A
2002/0015018	12/2001	Shimazu et al.	N/A	N/A
2002/0019584	12/2001	Schulze et al.	N/A	N/A
2002/0021700	12/2001	Hata et al.	N/A	N/A
2002/0026103	12/2001	Norris et al.	N/A	N/A
2002/0029776	12/2001	Blomquist	N/A	N/A
2002/0032583	12/2001	Joao	N/A	N/A
2002/0040208	12/2001	Flaherty et al.	N/A	N/A
2002/0040282	12/2001	Bailey et al.	N/A	N/A
2002/0044043	12/2001	Chaco et al.	N/A	N/A
2002/0044059	12/2001	Reeder et al.	N/A	N/A
2002/0082728	12/2001	Mueller et al.	N/A	N/A
2002/0087115	12/2001	Hartlaub	N/A	N/A
2002/0087116	12/2001	Hartlaub	N/A	N/A
2002/0095486	12/2001	Bahl	N/A	N/A
2002/0103675	12/2001	Vanelli	N/A	N/A
2002/0123905	12/2001	Goodroe et al.	N/A	N/A
2002/0143580	12/2001	Bristol et al.	N/A	N/A
2002/0152239	12/2001	Bautista-Lloyd et al.	N/A	N/A
2002/0154600	12/2001	Ido et al.	N/A	N/A
2002/0173702	12/2001	Lebel et al.	N/A	N/A
2002/0173875	12/2001	Wallace et al.	N/A	N/A
2002/0193679	12/2001	Malave et al.	N/A	N/A
2002/0194329	12/2001	Alling	N/A	N/A
2003/0009244	12/2002	Engleson	N/A	N/A
2003/0013959	12/2002	Grunwald et al.	N/A	N/A
2003/0014222	12/2002	Klass et al.	N/A	N/A
2003/0014817	12/2002	Gallant et al.	N/A	N/A
2003/0025602	12/2002	Medema et al.	N/A	N/A
2003/0028082	12/2002	Thompson	N/A	N/A
2003/0036683	12/2002	Kehr et al.	N/A	N/A

2003/0036744	12/2002	Struys et al.	N/A	N/A
2003/0047126	12/2002	Tomaschko	N/A	N/A
2003/0047600	12/2002	Nakanishi et al.	N/A	N/A
2003/0050621	12/2002	Lebel et al.	N/A	N/A
2003/0059750	12/2002	Bindler et al.	N/A	N/A
2003/0060688	12/2002	Ciarniello et al.	N/A	N/A
2003/0069963	12/2002	Jayant et al.	N/A	N/A
2003/0079746	12/2002	Hickle	N/A	N/A
2003/0097529	12/2002	Arimilli et al.	N/A	N/A
2003/0104982	12/2002	Wittmann et al.	N/A	N/A
2003/0105389	12/2002	Noonan et al.	N/A	N/A
2003/0106553	12/2002	Vanderveen	N/A	N/A
2003/0115358	12/2002	Yun	N/A	N/A
2003/0120384	12/2002	Haitin et al.	N/A	N/A
2003/0125662	12/2002	Bui	N/A	N/A
2003/0130616	12/2002	Steil	N/A	N/A
2003/0135087	12/2002	Hickle et al.	N/A	N/A
2003/0139701	12/2002	White et al.	N/A	N/A
2003/0140928	12/2002	Bui et al.	N/A	N/A
2003/0140929	12/2002	Wilkes et al.	N/A	N/A
2003/0141981	12/2002	Bui et al.	N/A	N/A
2003/0143746	12/2002	Sage, Jr.	N/A	N/A
2003/0144878	12/2002	Wilkes et al.	N/A	N/A
2003/0158749	12/2002	Olchanski et al.	N/A	N/A
2003/0187338	12/2002	Say et al.	N/A	N/A
2003/0200116	12/2002	Forrester	N/A	N/A
2003/0204416	12/2002	Acharya	N/A	N/A
2003/0204781	12/2002	Peebles et al.	N/A	N/A
2003/0212364	12/2002	Mann et al.	N/A	N/A
2003/0212379	12/2002	Bylund et al.	N/A	N/A
2003/0212821	12/2002	Gillies et al.	N/A	N/A
2003/0216831	12/2002	Hart et al.	N/A	N/A
2003/0217962	12/2002	Childers et al.	N/A	N/A
2004/0008123	12/2003	Carrender et al.	N/A	N/A
2004/0010786	12/2003	Cool et al.	N/A	N/A
2004/0015132	12/2003	Brown	N/A	N/A
2004/0019464	12/2003	Martucci et al.	N/A	N/A
2004/0019607	12/2003	Moubayed et al.	N/A	N/A
2004/0030323	12/2003	Ullestad et al.	N/A	N/A
2004/0039257	12/2003	Hickle	N/A	N/A
2004/0057226	12/2003	Berthou et al.	N/A	N/A
2004/0064341	12/2003	Langan et al.	N/A	N/A
2004/0064342	12/2003	Browne et al.	N/A	N/A
2004/0064435	12/2003	Moubayed et al.	N/A	N/A
2004/0073161	12/2003	Tachibana	N/A	N/A
2004/0073811	12/2003	Sanin	N/A	N/A
2004/0077934	12/2003	Massad	N/A	N/A
2004/0078231	12/2003	Wilkes et al.	N/A	N/A
2004/0078236	12/2003	Stoodley et al.	N/A	N/A
2004/0085186	12/2003	Eveland et al.	N/A	N/A
2004/0104271	12/2003	Martucci et al.	N/A	N/A
2004/0122530	12/2003	Hansen	N/A	N/A
2004/0128162	12/2003	Schlotterbeck et al.	N/A	N/A
2004/0128163	12/2003	Goodman et al.	N/A	N/A
2004/0133441	12/2003	Brady et al.	N/A	N/A
2004/0139004	12/2003	Cohen et al.	N/A	N/A
2004/0145480	12/2003	Despotis	N/A	N/A
2004/0147034	12/2003	Gore et al.	N/A	N/A
2004/0167464	12/2003	Ireland et al.	N/A	N/A
2004/0167465	12/2003	Kohler	N/A	N/A
2004/0167804	12/2003	Simpson	N/A	N/A
2004/0172222	12/2003	Simpson et al.	N/A	N/A
2004/0172283	12/2003	Vanderveen	N/A	N/A
2004/0172301	12/2003	Mihai et al.	N/A	N/A
2004/0172302	12/2003	Martucci et al.	N/A	N/A
2004/0176667	12/2003	Mihai et al.	N/A	N/A
2004/0176980	12/2003	Bulitta et al.	N/A	N/A
2004/0176984	12/2003	White et al.	N/A	N/A
2004/0181314	12/2003	Zaleski	N/A	N/A
2004/0189708	12/2003	Larcheveque et al.	N/A	N/A
2004/0193325	12/2003	Bonderud	N/A	N/A
2004/0193328	12/2003	Butterfield et al.	N/A	N/A
2004/0193453	12/2003	Butterfield et al.	N/A	N/A
2004/0204673	12/2003	Flaherty et al.	N/A	N/A
2004/0215278	12/2003	Stegink et al.	N/A	N/A

2004/0220517	12/2003	Starkweather et al.	N/A	N/A
2004/0225252	12/2003	Gillespie et al.	N/A	N/A
2004/0236240	12/2003	Kraus et al.	N/A	N/A
2004/0243438	12/2003	Mintz	N/A	N/A
2004/0254434	12/2003	Goodnow et al.	N/A	N/A
2005/0010269	12/2004	Lebel et al.	N/A	N/A
2005/0020886	12/2004	Hutchinson et al.	N/A	N/A
2005/0021006	12/2004	Tonnies	N/A	N/A
2005/0027560	12/2004	Cook	N/A	N/A
2005/0027567	12/2004	Taha	N/A	N/A
2005/0038311	12/2004	Kuth	N/A	N/A
2005/0038669	12/2004	Sachdeva et al.	N/A	N/A
2005/0038680	12/2004	McMahon	N/A	N/A
2005/0040226	12/2004	Al-Sheikh	N/A	N/A
2005/0043620	12/2004	Fallows et al.	N/A	N/A
2005/0049910	12/2004	Lancaster et al.	N/A	N/A
2005/0055242	12/2004	Bello et al.	N/A	N/A
2005/0055244	12/2004	Mullan et al.	N/A	N/A
2005/0065465	12/2004	Lebel et al.	N/A	N/A
2005/0065817	12/2004	Mihai et al.	N/A	N/A
2005/0075544	12/2004	Shapiro et al.	N/A	N/A
2005/0080801	12/2004	Kothandaraman et al.	N/A	N/A
2005/0086071	12/2004	Fox, Jr. et al.	N/A	N/A
2005/0086072	12/2004	Fox	N/A	N/A
2005/0088704	12/2004	Vaschillo et al.	N/A	N/A
2005/0090808	12/2004	Malave et al.	N/A	N/A
2005/0099624	12/2004	Staehr	N/A	N/A
2005/0102162	12/2004	Blumenfeld	N/A	N/A
2005/0102165	12/2004	Oshita et al.	N/A	N/A
2005/0102167	12/2004	Kapoor	N/A	N/A
2005/0102669	12/2004	Marney et al.	N/A	N/A
2005/0107923	12/2004	Vanderveen	N/A	N/A
2005/0108057	12/2004	Cohen et al.	N/A	N/A
2005/0117529	12/2004	Ramos-Escano	N/A	N/A
2005/0119788	12/2004	Engleson et al.	N/A	N/A
2005/0119914	12/2004	Batch	N/A	N/A
2005/0131739	12/2004	Rabinowitz et al.	N/A	N/A
2005/0135306	12/2004	McAllen et al.	N/A	N/A
2005/0137522	12/2004	Aoki	N/A	N/A
2005/0137573	12/2004	Mclaughlin	N/A	N/A
2005/0138428	12/2004	McAllen et al.	N/A	N/A
2005/0154769	12/2004	Eckart et al.	N/A	N/A
2005/0160057	12/2004	Wefers et al.	N/A	N/A
2005/0171503	12/2004	Van Den Berghe et al.	N/A	N/A
2005/0171815	12/2004	Vanderveen	N/A	N/A
2005/0177096	12/2004	Bollish et al.	N/A	N/A
2005/0177395	12/2004	Blomquist	N/A	N/A
2005/0182306	12/2004	Sloan	N/A	N/A
2005/0182355	12/2004	Bui	N/A	N/A
2005/0187950	12/2004	Parker	N/A	N/A
2005/0192557	12/2004	Brauker et al.	N/A	N/A
2005/0197554	12/2004	Polcha	N/A	N/A
2005/0197621	12/2004	Poulsen et al.	N/A	N/A
2005/0210037	12/2004	Wefers et al.	N/A	N/A
2005/0216479	12/2004	Wefers et al.	N/A	N/A
2005/0216480	12/2004	Wefers et al.	N/A	N/A
2005/0223045	12/2004	Funahashi et al.	N/A	N/A
2005/0224083	12/2004	Crass	N/A	N/A
2005/0234746	12/2004	Funahashi	N/A	N/A
2005/0240305	12/2004	Bogash et al.	N/A	N/A
2005/0246416	12/2004	Blomquist	N/A	N/A
2005/0251418	12/2004	Fox et al.	N/A	N/A
2005/0261660	12/2004	Choi	N/A	N/A
2005/0273059	12/2004	Mernoe et al.	N/A	N/A
2005/0273367	12/2004	Nourie et al.	N/A	N/A
2005/0277873	12/2004	Stewart et al.	N/A	N/A
2005/0277890	12/2004	Stewart et al.	N/A	N/A
2005/0277911	12/2004	Stewart et al.	N/A	N/A
2005/0278194	12/2004	Holland et al.	N/A	N/A
2006/0004772	12/2005	Hagan et al.	N/A	N/A
2006/0009727	12/2005	O'Mahony et al.	N/A	N/A
2006/0009734	12/2005	Martin	N/A	N/A
2006/0010098	12/2005	Goodnow et al.	N/A	N/A
2006/0042139	12/2005	Mendes	N/A	N/A
2006/0047270	12/2005	Shelton	N/A	N/A

2006/0053036	12/2005	Coffman et al.	N/A	N/A
2006/0064020	12/2005	Burnes et al.	N/A	N/A
2006/0074633	12/2005	Mahesh et al.	N/A	N/A
2006/0074920	12/2005	Wefers et al.	N/A	N/A
2006/0079831	12/2005	Gilbert	N/A	N/A
2006/0089539	12/2005	Miodownik et al.	N/A	N/A
2006/0089854	12/2005	Holland et al.	N/A	N/A
2006/0089855	12/2005	Holland et al.	N/A	N/A
2006/0100746	12/2005	Leibner-Druska	N/A	N/A
2006/0100907	12/2005	Holland et al.	N/A	N/A
2006/0106649	12/2005	Eggers et al.	N/A	N/A
2006/0111943	12/2005	Wu	N/A	N/A
2006/0116904	12/2005	Brem	N/A	N/A
2006/0116907	12/2005	Rhodes et al.	N/A	N/A
2006/0122481	12/2005	Sievenpiper et al.	N/A	N/A
2006/0122867	12/2005	Eggers et al.	N/A	N/A
2006/0129140	12/2005	Todd et al.	N/A	N/A
2006/0129429	12/2005	Moubayed et al.	N/A	N/A
2006/0129434	12/2005	Smitherman et al.	N/A	N/A
2006/0129435	12/2005	Smitherman et al.	N/A	N/A
2006/0136266	12/2005	Tarassenko et al.	N/A	N/A
2006/0136271	12/2005	Eggers et al.	N/A	N/A
2006/0143051	12/2005	Eggers et al.	N/A	N/A
2006/0173260	12/2005	Gaoni et al.	N/A	N/A
2006/0173406	12/2005	Hayes et al.	N/A	N/A
2006/0173715	12/2005	Wang et al.	N/A	N/A
2006/0173927	12/2005	Beyer et al.	N/A	N/A
2006/0190302	12/2005	Eggers et al.	N/A	N/A
2006/0195022	12/2005	Trepagnier et al.	N/A	N/A
2006/0200007	12/2005	Brockway et al.	N/A	N/A
2006/0200369	12/2005	Batch et al.	N/A	N/A
2006/0211404	12/2005	Cromp et al.	N/A	N/A
2006/0224141	12/2005	Rush et al.	N/A	N/A
2006/0229918	12/2005	Fotsch et al.	N/A	N/A
2006/0236373	12/2005	Graves et al.	N/A	N/A
2006/0247606	12/2005	Batch	N/A	N/A
2006/0253554	12/2005	Uwais	N/A	N/A
2006/0258985	12/2005	Russell	N/A	N/A
2006/0259327	12/2005	Hoag	N/A	N/A
2006/0264895	12/2005	Flanders	N/A	N/A
2006/0265246	12/2005	Hoag	N/A	N/A
2006/0267753	12/2005	Hussey et al.	N/A	N/A
2006/0268710	12/2005	Appanna et al.	N/A	N/A
2006/0270971	12/2005	Gelfand et al.	N/A	N/A
2006/0277206	12/2005	Bailey et al.	N/A	N/A
2006/0287885	12/2005	Frick	N/A	N/A
2007/0015972	12/2006	Wang et al.	N/A	N/A
2007/0016443	12/2006	Wachman et al.	N/A	N/A
2007/0021715	12/2006	Kohlbrenner et al.	N/A	N/A
2007/0027506	12/2006	Stender et al.	N/A	N/A
2007/0060796	12/2006	Kim	N/A	N/A
2007/0060870	12/2006	Tolle et al.	N/A	N/A
2007/0060871	12/2006	Istoc	N/A	N/A
2007/0061393	12/2006	Moore	N/A	N/A
2007/0065363	12/2006	Dalal et al.	N/A	N/A
2007/0073419	12/2006	Sesay	N/A	N/A
2007/0073822	12/2006	Bennett et al.	N/A	N/A
2007/0078314	12/2006	Grounsell	N/A	N/A
2007/0083870	12/2006	Kanakogi	N/A	N/A
2007/0088333	12/2006	Levin et al.	N/A	N/A
2007/0093786	12/2006	Goldsmith et al.	N/A	N/A
2007/0100665	12/2006	Brown	N/A	N/A
2007/0100667	12/2006	Bardy	N/A	N/A
2007/0106126	12/2006	Mannheimer et al.	N/A	N/A
2007/0112298	12/2006	Mueller et al.	N/A	N/A
2007/0116037	12/2006	Moore	N/A	N/A
2007/0118405	12/2006	Campbell et al.	N/A	N/A
2007/0135866	12/2006	Baker et al.	N/A	N/A
2007/0136098	12/2006	Smythe et al.	N/A	N/A
2007/0142822	12/2006	Remde	N/A	N/A
2007/0156282	12/2006	Dunn	N/A	N/A
2007/0156452	12/2006	Batch	N/A	N/A
2007/0169008	12/2006	Varanasi et al.	N/A	N/A
2007/0179448	12/2006	Lim et al.	N/A	N/A
2007/0186923	12/2006	Poutiatine et al.	N/A	N/A

2007/0191817	12/2006	Martin	N/A	N/A
2007/0191973	12/2006	Holzbauer et al.	N/A	N/A
2007/0213657	12/2006	Jennewine et al.	N/A	N/A
2007/0213684	12/2006	Hickle et al.	N/A	N/A
2007/0214003	12/2006	Holland et al.	N/A	N/A
2007/0215545	12/2006	Bissler et al.	N/A	N/A
2007/0229249	12/2006	McNeal et al.	N/A	N/A
2007/0232867	12/2006	Hansmann	N/A	N/A
2007/0233035	12/2006	Wehba et al.	N/A	N/A
2007/0233049	12/2006	Wehba et al.	N/A	N/A
2007/0233206	12/2006	Frikart	N/A	N/A
2007/0233520	12/2006	Wehba et al.	N/A	N/A
2007/0240215	12/2006	Flores	N/A	N/A
2007/0251835	12/2006	Mehta et al.	N/A	N/A
2007/0253021	12/2006	Mehta et al.	N/A	N/A
2007/0254593	12/2006	Jollota et al.	N/A	N/A
2007/0255125	12/2006	Moberg et al.	N/A	N/A
2007/0257788	12/2006	Carlson	N/A	N/A
2007/0258395	12/2006	Jollota et al.	N/A	N/A
2007/0299687	12/2006	Palmer et al.	N/A	N/A
2007/0299695	12/2006	Jung et al.	N/A	N/A
2008/0001771	12/2007	Faoro et al.	N/A	N/A
2008/0004904	12/2007	Tran	N/A	N/A
2008/0009684	12/2007	Corsetti et al.	N/A	N/A
2008/0033361	12/2007	Evans et al.	N/A	N/A
2008/0033966	12/2007	Wahl	N/A	N/A
2008/0034323	12/2007	Blomquist	N/A	N/A
2008/0041942	12/2007	Aissa	N/A	N/A
2008/0052704	12/2007	Wysocki	N/A	N/A
2008/0065007	12/2007	Peterson et al.	N/A	N/A
2008/0065417	12/2007	Jung et al.	N/A	N/A
2008/0071217	12/2007	Moubayed et al.	N/A	N/A
2008/0071251	12/2007	Moubayed et al.	N/A	N/A
2008/0086088	12/2007	Malcolm	N/A	N/A
2008/0091466	12/2007	Butler et al.	N/A	N/A
2008/0095339	12/2007	Elliott	N/A	N/A
2008/0097289	12/2007	Steil et al.	N/A	N/A
2008/0097552	12/2007	Dicks et al.	N/A	N/A
2008/0126969	12/2007	Blomquist	N/A	N/A
2008/0139907	12/2007	Rao et al.	N/A	N/A
2008/0148047	12/2007	Appenzeller et al.	N/A	N/A
2008/0149117	12/2007	Raghuram	N/A	N/A
2008/0154177	12/2007	Moubayed et al.	N/A	N/A
2008/0172337	12/2007	Banfield et al.	N/A	N/A
2008/0184219	12/2007	Matsumoto	N/A	N/A
2008/0188796	12/2007	Steil et al.	N/A	N/A
2008/0214919	12/2007	Harmon et al.	N/A	N/A
2008/0246748	12/2007	Cassidy et al.	N/A	N/A
2008/0256305	12/2007	Kwon	N/A	N/A
2008/0259926	12/2007	Tavares et al.	N/A	N/A
2008/0262469	12/2007	Bristol et al.	N/A	N/A
2008/0269714	12/2007	Mastrototaro et al.	N/A	N/A
2008/0269723	12/2007	Mastrototaro et al.	N/A	N/A
2008/0275384	12/2007	Mastrototaro et al.	N/A	N/A
2008/0300572	12/2007	Rankers et al.	N/A	N/A
2008/0301298	12/2007	Bernardi et al.	N/A	N/A
2008/0320387	12/2007	Sasaki et al.	N/A	N/A
2008/0320466	12/2007	Dias	N/A	N/A
2009/0003554	12/2008	Katis et al.	N/A	N/A
2009/0005703	12/2008	Fasciano	N/A	N/A
2009/0005728	12/2008	Weinert et al.	N/A	N/A
2009/0006061	12/2008	Thukral et al.	N/A	N/A
2009/0006129	12/2008	Thukral	N/A	N/A
2009/0006133	12/2008	Weinert	N/A	N/A
2009/0018495	12/2008	Panduro	N/A	N/A
2009/0036750	12/2008	Weinstein et al.	N/A	N/A
2009/0051560	12/2008	Manning et al.	N/A	N/A
2009/0054743	12/2008	Stewart	N/A	N/A
2009/0054754	12/2008	McMahon et al.	N/A	N/A
2009/0057399	12/2008	Sajkowsky	N/A	N/A
2009/0063187	12/2008	Johnson et al.	N/A	N/A
2009/0069785	12/2008	Miller et al.	N/A	N/A
2009/0099867	12/2008	Newman	N/A	N/A
2009/0135196	12/2008	Holland et al.	N/A	N/A
2009/0143662	12/2008	Estes et al.	N/A	N/A

2009/0149743	12/2008	Barron et al.	N/A	N/A
2009/0150174	12/2008	Buck et al.	N/A	N/A
2009/0150439	12/2008	Gejdos et al.	N/A	N/A
2009/0150878	12/2008	Pathak et al.	N/A	N/A
2009/0156991	12/2008	Roberts	N/A	N/A
2009/0157695	12/2008	Roberts	N/A	N/A
2009/0158274	12/2008	Roberts	N/A	N/A
2009/0177146	12/2008	Nesbitt et al.	N/A	N/A
2009/0177769	12/2008	Roberts	N/A	N/A
2009/0177992	12/2008	Rubalcaba et al.	N/A	N/A
2009/0183147	12/2008	Davis et al.	N/A	N/A
2009/0209938	12/2008	Aalto-Setala	N/A	N/A
2009/0210250	12/2008	Prax et al.	N/A	N/A
2009/0221890	12/2008	Saffer et al.	N/A	N/A
2009/0231249	12/2008	Wang et al.	N/A	N/A
2009/0270833	12/2008	DeBelser	N/A	N/A
2009/0275886	12/2008	Blomquist et al.	N/A	N/A
2009/0275896	12/2008	Kamen et al.	N/A	N/A
2009/0284691	12/2008	Marhefka et al.	N/A	N/A
2009/0292340	12/2008	Mass et al.	N/A	N/A
2009/0306573	12/2008	Gagner et al.	N/A	N/A
2009/0326340	12/2008	Wang	N/A	N/A
2009/0326516	12/2008	Bangera et al.	N/A	N/A
2010/0008377	12/2009	Hasti et al.	N/A	N/A
2010/0022988	12/2009	Wochner	N/A	N/A
2010/0036310	12/2009	Hillman	N/A	N/A
2010/0056992	12/2009	Hayter	N/A	N/A
2010/0083060	12/2009	Rahman	N/A	N/A
2010/0095229	12/2009	Dixon et al.	N/A	N/A
2010/0121170	12/2009	Rule	N/A	N/A
2010/0121246	12/2009	Peters et al.	N/A	N/A
2010/0121415	12/2009	Skelton et al.	N/A	N/A
2010/0121654	12/2009	Portnoy et al.	N/A	N/A
2010/0121752	12/2009	Banigan et al.	N/A	N/A
2010/0130933	12/2009	Holland et al.	N/A	N/A
2010/0131434	12/2009	Magent et al.	N/A	N/A
2010/0138523	12/2009	Umess et al.	N/A	N/A
2010/0146137	12/2009	Wu et al.	N/A	N/A
2010/0156633	12/2009	Buck et al.	N/A	N/A
2010/0160854	12/2009	Gauthier	N/A	N/A
2010/0160860	12/2009	Celentano et al.	N/A	N/A
2010/0174266	12/2009	Estes	N/A	N/A
2010/0191525	12/2009	Rabenko et al.	N/A	N/A
2010/0198034	12/2009	Thomas et al.	N/A	N/A
2010/0198196	12/2009	Wei	N/A	N/A
2010/0200506	12/2009	Ware et al.	N/A	N/A
2010/0209268	12/2009	Davis	N/A	N/A
2010/0212675	12/2009	Walling et al.	N/A	N/A
2010/0217621	12/2009	Schoenberg	N/A	N/A
2010/0234708	12/2009	Buck et al.	N/A	N/A
2010/0250732	12/2009	Bucknell	N/A	N/A
2010/0271479	12/2009	Heydlauf	N/A	N/A
2010/0273738	12/2009	Valcke et al.	N/A	N/A
2010/0274218	12/2009	Yodfat et al.	N/A	N/A
2010/0280486	12/2009	Khair et al.	N/A	N/A
2010/0292634	12/2009	Kircher	N/A	N/A
2010/0298765	12/2009	Budiman et al.	N/A	N/A
2010/0318025	12/2009	John	N/A	N/A
2011/0001605	12/2010	Kiani et al.	N/A	N/A
2011/0021898	12/2010	Wei et al.	N/A	N/A
2011/0028885	12/2010	Eggers et al.	N/A	N/A
2011/0040158	12/2010	Katz et al.	N/A	N/A
2011/0060758	12/2010	Schlotterbeck et al.	N/A	N/A
2011/0071844	12/2010	Cannon et al.	N/A	N/A
2011/0072379	12/2010	Gannon	N/A	N/A
2011/0078253	12/2010	Chan et al.	N/A	N/A
2011/0078608	12/2010	Gannon et al.	N/A	N/A
2011/0093284	12/2010	Dicks et al.	N/A	N/A
2011/0099313	12/2010	Bolanowski	N/A	N/A
2011/0106318	12/2010	Ledford et al.	N/A	N/A
2011/0125095	12/2010	Lebel et al.	N/A	N/A
2011/0138185	12/2010	Ju et al.	N/A	N/A
2011/0166628	12/2010	Jain	N/A	N/A
2011/0175728	12/2010	Baker, Jr.	N/A	N/A
2011/0178462	12/2010	Moberg et al.	N/A	N/A

2011/0185010	12/2010	Shatsky et al.	N/A	N/A
2011/0196748	12/2010	Caron et al.	N/A	N/A
2011/0231216	12/2010	Fyke et al.	N/A	N/A
2011/0252230	12/2010	Segre et al.	N/A	N/A
2011/0257496	12/2010	Terashima et al.	N/A	N/A
2011/0257798	12/2010	Ali et al.	N/A	N/A
2011/0259954	12/2010	Bartz et al.	N/A	N/A
2011/0264043	12/2010	Kotnick et al.	N/A	N/A
2011/0264044	12/2010	Bartz et al.	N/A	N/A
2011/0266221	12/2010	Ware et al.	N/A	N/A
2011/0270045	12/2010	Lebel et al.	N/A	N/A
2011/0275904	12/2010	Lebel et al.	N/A	N/A
2011/0286457	12/2010	Ee	N/A	N/A
2011/0289162	12/2010	Furlong	N/A	N/A
2011/0289314	12/2010	Whitcomb	N/A	N/A
2011/0289497	12/2010	Kiaie et al.	N/A	N/A
2011/0295196	12/2010	Chazot et al.	N/A	N/A
2011/0295341	12/2010	Estes et al.	N/A	N/A
2011/0296051	12/2010	Vange	N/A	N/A
2011/0296411	12/2010	Tang et al.	N/A	N/A
2011/0313789	12/2010	Karmen et al.	N/A	N/A
2011/0319813	12/2010	Kamen et al.	N/A	N/A
2011/0320049	12/2010	Chossat et al.	N/A	N/A
2012/0005680	12/2011	Dolby et al.	N/A	N/A
2012/0011253	12/2011	Friedman et al.	N/A	N/A
2012/0016305	12/2011	Jollota	N/A	N/A
2012/0029941	12/2011	Malave et al.	N/A	N/A
2012/0036102	12/2011	Fletcher et al.	N/A	N/A
2012/0036550	12/2011	Rodriguez	N/A	N/A
2012/0066501	12/2011	Xiong	N/A	N/A
2012/0070045	12/2011	Vesper et al.	N/A	N/A
2012/0079084	12/2011	Forssell et al.	N/A	N/A
2012/0095437	12/2011	Hemmerling	N/A	N/A
2012/0112903	12/2011	Kaib et al.	N/A	N/A
2012/0130198	12/2011	Beaule	N/A	N/A
2012/0143116	12/2011	Ware et al.	N/A	N/A
2012/0150556	12/2011	Galasso et al.	N/A	N/A
2012/0157920	12/2011	Flachbart et al.	N/A	N/A
2012/0179135	12/2011	Rinehart et al.	N/A	N/A
2012/0179136	12/2011	Rinehart et al.	N/A	N/A
2012/0185267	12/2011	Kamen et al.	N/A	N/A
2012/0203177	12/2011	Lanier	N/A	N/A
2012/0245554	12/2011	Kawamura	N/A	N/A
2012/0259978	12/2011	Petersen et al.	N/A	N/A
2012/0260012	12/2011	Gao-Saari et al.	N/A	N/A
2012/0277716	12/2011	Ali et al.	N/A	N/A
2012/0283630	12/2011	Lee et al.	N/A	N/A
2012/0284734	12/2011	McQuaid et al.	N/A	N/A
2012/0323212	12/2011	Murphy	N/A	N/A
2012/0330380	12/2011	Corndorf	N/A	N/A
2013/0006666	12/2012	Schneider	N/A	N/A
2013/0006702	12/2012	Wu	N/A	N/A
2013/0012877	12/2012	Debelser et al.	N/A	N/A
2013/0012879	12/2012	Debelser et al.	N/A	N/A
2013/0012880	12/2012	Blomquist	N/A	N/A
2013/0015980	12/2012	Evans et al.	N/A	N/A
2013/0036403	12/2012	Geist	N/A	N/A
2013/0036412	12/2012	Birtwhistle et al.	N/A	N/A
2013/0066265	12/2012	Grant	N/A	N/A
2013/0072872	12/2012	Yodfat et al.	N/A	N/A
2013/0091350	12/2012	Gluck	N/A	N/A
2013/0096444	12/2012	Condurso et al.	N/A	N/A
2013/0096648	12/2012	Benson	N/A	N/A
2013/0102963	12/2012	Marsh et al.	N/A	N/A
2013/0114594	12/2012	Van Zijst	N/A	N/A
2013/0116578	12/2012	An	N/A	N/A
2013/0133083	12/2012	Kurumai	N/A	N/A
2013/0138452	12/2012	Cork et al.	N/A	N/A
2013/0144206	12/2012	Lee et al.	N/A	N/A
2013/0150824	12/2012	Estes et al.	N/A	N/A
2013/0158504	12/2012	Ruchti et al.	N/A	N/A
2013/0167245	12/2012	Birtwhistle et al.	N/A	N/A
2013/0173473	12/2012	Birtwhistle et al.	N/A	N/A
2013/0191770	12/2012	Bartz et al.	N/A	N/A
2013/0204188	12/2012	Kamen et al.	N/A	N/A

2013/0218080	12/2012	Peterfreund et al.	N/A	N/A
2013/0261993	12/2012	Ruchti et al.	N/A	N/A
2013/0274669	12/2012	Stempfle et al.	N/A	N/A
2013/0275539	12/2012	Gross et al.	N/A	N/A
2013/0291116	12/2012	Homer	N/A	N/A
2013/0296823	12/2012	Melker et al.	N/A	N/A
2013/0296984	12/2012	Burnett et al.	N/A	N/A
2013/0317753	12/2012	Kamen et al.	N/A	N/A
2013/0346108	12/2012	Kamen et al.	N/A	N/A
2014/0025392	12/2013	Chandrasenan	N/A	N/A
2014/0039446	12/2013	Day	N/A	N/A
2014/0108783	12/2013	Suzuki	N/A	N/A
2014/0142540	12/2013	Imhof	N/A	N/A
2014/0142963	12/2013	Hill et al.	N/A	N/A
2014/0163517	12/2013	Finan et al.	N/A	N/A
2014/0172994	12/2013	Raumann et al.	N/A	N/A
2014/0180711	12/2013	Kamen et al.	N/A	N/A
2014/0197950	12/2013	Shupp et al.	N/A	N/A
2014/0215490	12/2013	Mathur et al.	N/A	N/A
2014/0257251	12/2013	Bush et al.	N/A	N/A
2014/0266790	12/2013	Al-Ali et al.	N/A	N/A
2014/0266794	12/2013	Brown et al.	N/A	N/A
2014/0269643	12/2013	Sun	N/A	N/A
2014/0276571	12/2013	Ludolph	N/A	N/A
2014/0280522	12/2013	Watte	N/A	N/A
2014/0294177	12/2013	Shastry et al.	N/A	N/A
2014/0297329	12/2013	Rock	N/A	N/A
2014/0316819	12/2013	Dunsirn et al.	N/A	N/A
2014/0318639	12/2013	Peret et al.	N/A	N/A
2014/0350513	12/2013	Oruklu et al.	N/A	N/A
2014/0358077	12/2013	Oruklu et al.	N/A	N/A
2014/0366878	12/2013	Baron	N/A	N/A
2014/0371543	12/2013	Steinhauer et al.	N/A	N/A
2015/0005935	12/2014	Bae et al.	N/A	N/A
2015/0006907	12/2014	Brouwer et al.	N/A	N/A
2015/0045729	12/2014	Denzer et al.	N/A	N/A
2015/0058044	12/2014	Butler et al.	N/A	N/A
2015/0058960	12/2014	Schmoyer et al.	N/A	N/A
2015/0066531	12/2014	Jacobson et al.	N/A	N/A
2015/0081894	12/2014	Blomquist	N/A	N/A
2015/0100038	12/2014	McCann et al.	N/A	N/A
2015/0100787	12/2014	Westin et al.	N/A	N/A
2015/0117234	12/2014	Raman et al.	N/A	N/A
2015/0134265	12/2014	Kohlbrecher et al.	N/A	N/A
2015/0141955	12/2014	Ruchti et al.	N/A	N/A
2015/0151051	12/2014	Tsoukalis	N/A	N/A
2015/0161354	12/2014	Blomquist	N/A	N/A
2015/0199192	12/2014	Borges et al.	N/A	N/A
2015/0199485	12/2014	Borges et al.	N/A	N/A
2015/0207626	12/2014	Neftel et al.	N/A	N/A
2015/0220890	12/2014	Seshadri et al.	N/A	N/A
2015/0230760	12/2014	Schneider	N/A	N/A
2015/0281128	12/2014	Sindhu	N/A	N/A
2015/0325064	12/2014	Downey	N/A	N/A
2015/0328396	12/2014	Adams et al.	N/A	N/A
2015/0352301	12/2014	Stedman et al.	N/A	N/A
2015/0371004	12/2014	Jones	N/A	N/A
2015/0379237	12/2014	Mills et al.	N/A	N/A
2016/0001003	12/2015	Perazzo et al.	N/A	N/A
2016/0006695	12/2015	Prodoehl et al.	N/A	N/A
2016/0015885	12/2015	Pananen et al.	N/A	N/A
2016/0034655	12/2015	Gray et al.	N/A	N/A
2016/0045661	12/2015	Gray et al.	N/A	N/A
2016/0051749	12/2015	Istoc	N/A	N/A
2016/0051751	12/2015	Silkaitis et al.	N/A	N/A
2016/0063471	12/2015	Kobres et al.	N/A	N/A
2016/0103960	12/2015	Hume et al.	N/A	N/A
2016/0158437	12/2015	Biasi et al.	N/A	N/A
2016/0228633	12/2015	Welsch et al.	N/A	N/A
2016/0241391	12/2015	Fenster	N/A	N/A
2016/0277152	12/2015	Xiang et al.	N/A	N/A
2016/0285876	12/2015	Perez et al.	N/A	N/A
2016/0317742	12/2015	Gannon et al.	N/A	N/A
2016/0350513	12/2015	Jacobson et al.	N/A	N/A
2016/0378618	12/2015	Cmielowski	N/A	N/A

2017/0024534	12/2016	Arrizza et al.	N/A	N/A
2017/0034277	12/2016	Jackson et al.	N/A	N/A
2017/0063559	12/2016	Wallrabenstein	N/A	N/A
2017/0099148	12/2016	Ochmanski et al.	N/A	N/A
2017/0104645	12/2016	Wooton et al.	N/A	N/A
2017/0111301	12/2016	Robinson	N/A	N/A
2017/0140134	12/2016	Brough et al.	N/A	N/A
2017/0146381	12/2016	Eckel et al.	N/A	N/A
2017/0149567	12/2016	Moskal	N/A	N/A
2017/0149929	12/2016	Moskal	N/A	N/A
2017/0214762	12/2016	Swain et al.	N/A	N/A
2017/0246388	12/2016	Kohlbrecher	N/A	N/A
2017/0258401	12/2016	Volpe	N/A	N/A
2017/0258986	12/2016	Tsoiukalis	N/A	N/A
2017/0262590	12/2016	Karakosta et al.	N/A	N/A
2017/0274140	12/2016	Howard et al.	N/A	N/A
2017/0286637	12/2016	Arrizza et al.	N/A	N/A
2017/0319780	12/2016	Belkin et al.	N/A	N/A
2017/0325091	12/2016	Freeman et al.	N/A	N/A
2017/0331804	12/2016	Jellison et al.	N/A	N/A
2017/0351841	12/2016	Moskal	N/A	N/A
2018/0008772	12/2017	Wehba et al.	N/A	N/A
2018/0028742	12/2017	Day et al.	N/A	N/A
2018/0043094	12/2017	Day et al.	N/A	N/A
2018/0063724	12/2017	Zhang et al.	N/A	N/A
2018/0121613	12/2017	Connely, IV et al.	N/A	N/A
2018/0122502	12/2017	Jones et al.	N/A	N/A
2018/0126067	12/2017	Ledford et al.	N/A	N/A
2018/0157821	12/2017	Fan	N/A	N/A
2018/0181712	12/2017	Ensey et al.	N/A	N/A
2018/0247712	12/2017	Muhsin et al.	N/A	N/A
2018/0272117	12/2017	Fangrow	N/A	N/A
2018/0278594	12/2017	Schiffman et al.	N/A	N/A
2018/0317826	12/2017	Muhsin et al.	N/A	N/A
2018/0322948	12/2017	Drost et al.	N/A	N/A
2018/0359085	12/2017	Dervyn	N/A	N/A
2019/0006044	12/2018	Brask	N/A	N/A
2019/0030329	12/2018	Hannaman et al.	N/A	N/A
2019/0036688	12/2018	Wasily et al.	N/A	N/A
2019/0096518	12/2018	Pace	N/A	N/A
2019/0132196	12/2018	Trivedi et al.	N/A	N/A
2019/0147998	12/2018	Ruchti et al.	N/A	N/A
2019/0166501	12/2018	Debates et al.	N/A	N/A
2019/0172590	12/2018	Vesto et al.	N/A	N/A
2019/0207965	12/2018	Espinosa	N/A	N/A
2019/0228863	12/2018	Dharwad et al.	N/A	N/A
2019/0229982	12/2018	Ikuta et al.	N/A	N/A
2019/0240405	12/2018	Wehba et al.	N/A	N/A
2019/0243829	12/2018	Butler et al.	N/A	N/A
2019/0244689	12/2018	Atkin	N/A	N/A
2019/0245942	12/2018	Moskal	N/A	N/A
2019/0269852	12/2018	Kohlbrecher	N/A	N/A
2019/0311803	12/2018	Kohlbrecher et al.	N/A	N/A
2019/0348160	12/2018	Heavelyn et al.	N/A	N/A
2019/0392929	12/2018	Gassman	N/A	N/A
2020/0023127	12/2019	Simpson et al.	N/A	N/A
2020/0027541	12/2019	Xavier et al.	N/A	N/A
2020/0027548	12/2019	Xavier et al.	N/A	N/A
2020/0027550	12/2019	Xavier et al.	N/A	N/A
2020/0027551	12/2019	Xavier et al.	N/A	N/A
2020/0035355	12/2019	Xavier et al.	N/A	N/A
2020/0054825	12/2019	Kamen et al.	N/A	N/A
2020/0118692	12/2019	Booker et al.	N/A	N/A
2020/0153627	12/2019	Wentz	N/A	N/A
2020/0206413	12/2019	Silkaitis et al.	N/A	N/A
2020/0220865	12/2019	Finger et al.	N/A	N/A
2020/0282139	12/2019	Susi	N/A	N/A
2020/0334497	12/2019	Barrett et al.	N/A	N/A
2020/0335194	12/2019	Jacobson et al.	N/A	N/A
2020/0351376	12/2019	Moskal	N/A	N/A
2020/0353167	12/2019	Vivek et al.	N/A	N/A
2020/0353168	12/2019	Keenan et al.	N/A	N/A
2021/0014259	12/2020	Harris et al.	N/A	N/A
2021/0043296	12/2020	Xavier et al.	N/A	N/A
2021/0045640	12/2020	Poltorak	N/A	N/A

2021/0085855	12/2020	Belkin et al.	N/A	N/A
2021/0098106	12/2020	Kohlbrecher et al.	N/A	N/A
2021/0098107	12/2020	Xavier et al.	N/A	N/A
2021/0252210	12/2020	Day et al.	N/A	N/A
2021/0358603	12/2020	Xavier et al.	N/A	N/A
2021/0375421	12/2020	Ruchti et al.	N/A	N/A
2021/0375438	12/2020	Xavier et al.	N/A	N/A
2021/0409362	12/2020	Katis et al.	N/A	N/A
2022/0023535	12/2021	Day	N/A	N/A
2022/0037011	12/2021	Fryman	N/A	N/A
2022/0037012	12/2021	Fryman	N/A	N/A
2022/0051777	12/2021	Xavier et al.	N/A	N/A
2022/0062541	12/2021	Kamen et al.	N/A	N/A
2022/0129452	12/2021	Butler et al.	N/A	N/A
2022/0139536	12/2021	Xavier et al.	N/A	N/A
2022/0139537	12/2021	Xavier et al.	N/A	N/A
2022/0139538	12/2021	Xavier et al.	N/A	N/A
2022/0150307	12/2021	Walsh et al.	N/A	N/A
2022/0165404	12/2021	Vivek et al.	N/A	N/A
2022/0189605	12/2021	Kelly et al.	N/A	N/A
2022/0223283	12/2021	Biasi et al.	N/A	N/A
2022/0270736	12/2021	Kohlbrecher	N/A	N/A
2022/0328175	12/2021	Arrizza et al.	N/A	N/A
2022/0331513	12/2021	Howard et al.	N/A	N/A
2022/0344023	12/2021	Xavier et al.	N/A	N/A
2022/0375565	12/2021	Xavier et al.	N/A	N/A
2022/0384059	12/2021	Xavier et al.	N/A	N/A
2023/0009405	12/2022	Xavier et al.	N/A	N/A
2023/0009417	12/2022	Xavier et al.	N/A	N/A
2023/0139360	12/2022	Kohlbrecher et al.	N/A	N/A
2023/0145267	12/2022	Xavier et al.	N/A	N/A
2023/0147762	12/2022	Xavier et al.	N/A	N/A
2023/0166026	12/2022	Jacobson et al.	N/A	N/A
2023/0253108	12/2022	Dharwad et al.	N/A	N/A
2023/0285660	12/2022	Day et al.	N/A	N/A
2023/0298768	12/2022	Jacobson et al.	N/A	N/A
2023/0320935	12/2022	Tagliamento	N/A	N/A
2023/0321350	12/2022	Day et al.	N/A	N/A
2023/0321351	12/2022	Wehba et al.	N/A	N/A
2023/0326570	12/2022	Kohlbrecher	N/A	N/A
2023/0410989	12/2022	Xavier et al.	N/A	N/A
2024/0038358	12/2023	Xavier et al.	N/A	N/A
2024/0047035	12/2023	Ruchti et al.	N/A	N/A
2024/0071609	12/2023	Rohlwing	N/A	N/A

FOREIGN PATENT DOCUMENTS

Patent No.	Application Date	Country	CPC
2004226440	12/2003	AU	N/A
2004305087	12/2004	AU	N/A
2 060 151	12/1996	CA	N/A
2 125 300	12/1998	CA	N/A
2 630 102	12/2007	CA	N/A
2 687 587	12/2007	CA	N/A
2 897 897	12/2013	CA	N/A
2 898 825	12/2013	CA	N/A
2 900 564	12/2013	CA	N/A
2 606 968	12/2019	CA	N/A
1759398	12/2005	CN	N/A
102521474	12/2011	CN	N/A
103816582	12/2013	CN	N/A
103920206	12/2013	CN	N/A
102300501	12/2014	CN	N/A
104487976	12/2014	CN	N/A
107810536	12/2022	CN	N/A
01110843	12/2002	CO	N/A
31 12 762	12/1982	DE	N/A
34 35 647	12/1984	DE	N/A
198 44 252	12/1999	DE	N/A
199 32 147	12/2000	DE	N/A
103 52 456	12/2004	DE	N/A
0 319 267	12/1988	EP	N/A
0 380 061	12/1989	EP	N/A
0 384 155	12/1989	EP	N/A
0 460 533	12/1990	EP	N/A
0 564 127	12/1992	EP	N/A

0 633 035	12/1994	EP	N/A
0 652 528	12/1994	EP	N/A
0 672 427	12/1994	EP	N/A
0 683 465	12/1994	EP	N/A
0 880 936	12/1997	EP	N/A
1 050 993	12/1999	EP	N/A
1 157 711	12/2000	EP	N/A
1 174 817	12/2001	EP	N/A
0 664 102	12/2001	EP	N/A
1 197 178	12/2001	EP	N/A
0 830 775	12/2001	EP	N/A
1 500 025	12/2002	EP	N/A
1 487 171	12/2006	EP	N/A
1 933 497	12/2007	EP	N/A
2 026 223	12/2008	EP	N/A
2 113 842	12/2008	EP	N/A
2 228 004	12/2009	EP	N/A
2 243 506	12/2009	EP	N/A
2 410 448	12/2011	EP	N/A
2 742 961	12/2013	EP	N/A
2 874 087	12/2014	EP	N/A
2 371 995	12/2011	ES	N/A
2 717 919	12/1994	FR	N/A
2 285 135	12/1994	GB	N/A
04-161139	12/1991	JP	N/A
07-502678	12/1994	JP	N/A
11-500643	12/1998	JP	N/A
2000-316820	12/1999	JP	N/A
2002-531154	12/2001	JP	N/A
2003-016183	12/2002	JP	N/A
2003-296173	12/2002	JP	N/A
2003-308586	12/2002	JP	N/A
2005-021463	12/2004	JP	N/A
2005-527284	12/2004	JP	N/A
2005-284846	12/2004	JP	N/A
2006-047319	12/2005	JP	N/A
2006-520949	12/2005	JP	N/A
2007-518479	12/2006	JP	N/A
2007-525256	12/2006	JP	N/A
2008-080036	12/2007	JP	N/A
2008-516303	12/2007	JP	N/A
2008-158622	12/2007	JP	N/A
2008-529675	12/2007	JP	N/A
2009-163534	12/2008	JP	N/A
2010-502361	12/2009	JP	N/A
2011-506048	12/2010	JP	N/A
2012-011204	12/2011	JP	N/A
2012-070991	12/2011	JP	N/A
2012-523895	12/2011	JP	N/A
2014-068283	12/2013	JP	N/A
5647644	12/2014	JP	N/A
200426656	12/2003	TW	N/A
1631966	12/2017	TW	N/A
WO 84/001719	12/1983	WO	N/A
WO 91/016416	12/1990	WO	N/A
WO 92/010985	12/1991	WO	N/A
WO 92/013322	12/1991	WO	N/A
WO 94/005355	12/1993	WO	N/A
WO 96/008755	12/1995	WO	N/A
WO 96/025186	12/1995	WO	N/A
WO 96/025963	12/1995	WO	N/A
WO 98/012670	12/1997	WO	N/A
WO 98/019263	12/1997	WO	N/A
WO 99/051003	12/1998	WO	N/A
WO 00/013580	12/1999	WO	N/A
WO 00/053243	12/1999	WO	N/A
WO 01/014974	12/2000	WO	N/A
WO 01/033484	12/2000	WO	N/A
WO 01/045014	12/2000	WO	N/A
WO 01/083007	12/2000	WO	N/A
WO 02/005702	12/2001	WO	N/A
WO 02/036044	12/2001	WO	N/A
WO 02/049153	12/2001	WO	N/A
WO 02/049279	12/2001	WO	N/A
WO 02/069099	12/2001	WO	N/A

WO 02/081015	12/2001	WO	N/A
WO 02/088875	12/2001	WO	N/A
WO 03/006091	12/2002	WO	N/A
WO 03/023551	12/2002	WO	N/A
WO 03/050917	12/2002	WO	N/A
WO 03/091836	12/2002	WO	N/A
WO 03/094092	12/2002	WO	N/A
WO 2004/060455	12/2003	WO	N/A
WO 2004/070557	12/2003	WO	N/A
WO 2004/070562	12/2003	WO	N/A
WO 2004/072828	12/2003	WO	N/A
WO 2005/036447	12/2004	WO	N/A
WO 2005/050526	12/2004	WO	N/A
WO 2005/057175	12/2004	WO	N/A
WO 2005/066872	12/2004	WO	N/A
WO 2007/087443	12/2006	WO	N/A
WO 2007/117705	12/2006	WO	N/A
WO 2007/127879	12/2006	WO	N/A
WO 2007/127880	12/2006	WO	N/A
WO 2008/059495	12/2007	WO	N/A
WO 2008/064254	12/2007	WO	N/A
WO 2008/067245	12/2007	WO	N/A
WO 2008/082854	12/2007	WO	N/A
WO 2008/088490	12/2007	WO	N/A
WO 2008/097316	12/2007	WO	N/A
WO 2008/103915	12/2007	WO	N/A
WO 2008/124478	12/2007	WO	N/A
WO 2008/134146	12/2007	WO	N/A
WO 2009/016504	12/2008	WO	N/A
WO 2009/023406	12/2008	WO	N/A
WO 2009/023407	12/2008	WO	N/A
WO 2009/023634	12/2008	WO	N/A
WO 2009/036327	12/2008	WO	N/A
WO 2009/049252	12/2008	WO	N/A
WO 2010/017279	12/2009	WO	N/A
WO 2010/033919	12/2009	WO	N/A
WO 2010/053703	12/2009	WO	N/A
WO 2010/075371	12/2009	WO	N/A
WO 2010/099313	12/2009	WO	N/A
WO 2010/114929	12/2009	WO	N/A
WO 2010/119409	12/2009	WO	N/A
WO 2010/124127	12/2009	WO	N/A
WO 2010/130992	12/2009	WO	N/A
WO 2010/135646	12/2009	WO	N/A
WO 2010/135654	12/2009	WO	N/A
WO 2010/135686	12/2009	WO	N/A
WO 2011/005633	12/2010	WO	N/A
WO 2011/022549	12/2010	WO	N/A
WO 2012/048833	12/2011	WO	N/A
WO 2012/049214	12/2011	WO	N/A
WO 2012/049218	12/2011	WO	N/A
WO 2012/120078	12/2011	WO	N/A
WO 2012/140547	12/2011	WO	N/A
WO 2012/164556	12/2011	WO	N/A
WO 2012/170942	12/2011	WO	N/A
WO 2013/045506	12/2012	WO	N/A
WO 2014/100736	12/2013	WO	N/A
WO 2014/131729	12/2013	WO	N/A
WO 2014/131730	12/2013	WO	N/A
WO 2015/047595	12/2014	WO	N/A
WO 2015/124569	12/2014	WO	N/A
WO 2016/179389	12/2015	WO	N/A
WO 2017/176928	12/2016	WO	N/A
WO 2017/200989	12/2016	WO	N/A
WO 2019/219290	12/2018	WO	N/A
WO 00/003344	12/2019	WO	N/A
WO 2020/227403	12/2019	WO	N/A
WO 2021/201884	12/2020	WO	N/A
WO 2022/006014	12/2021	WO	N/A
WO 2022/051230	12/2021	WO	N/A
WO 2023/159134	12/2022	WO	N/A

OTHER PUBLICATIONS

Sethia et al., “Security Framework for Portable NFC Mobile Based Health Record System”, Oct. 2016, IEEE 12th International Conference on Wireless and M
Ahn et al., “Towards Scalable Authentication in Health Services”, Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collab

Akridge, Jeannie, “New Pumps Outsmart User Error”, Healthcare Purchasing News, Apr. 2011, pp. 10, <http://web.archive.org/web/20110426122450/http://w
Alur et al., “Formal Specifications and Analysis of the Computer-Assisted Resuscitation Algorithm (CARA) Infusion Pump Control System”, International Jo
319. cited by applicant
Aragon, Daleen RN, Ph.D., CCRN, “Evaluation of Nursing Work Effort and Perceptions About Blood Glucose Testing in Tight Glycemic Control”, American
ASHP Advantage, “Improving Medication Safety in Health Systems Through Innovations in Automation Technology”, Proceedings of Educational Symposiu
9, 2004, Orlando, FL, pp. 28. cited by applicant
Beard et al., “Total Quality Pain Management: History, Background, Resources”, Abbott Laboratories, TQPM Survey History, available Feb. 2015 or earlier, p
Bektas et al., “Bluetooth Communication Employing Antenna Diversity”, Proceedings of Eight IEEE International Symposium on Computers and Communic
Bellare et al., “Security Proofs for Identity-Based Identification and Signature Schemes”, Lecture Notes in Computer Science, Jan. 2009, vol. 22, No. 1, pp. 18
Bequette, Ph.D., “A Critical Assessment of Algorithms and Challenges in the Development of a Closed-Loop Artificial Pancreas”, Diabetes Technology & The
Bequette, B. Wayne, Ph.D., “Analysis of Algorithms for Intensive Care Unit Blood Glucose Control”, Journal of Diabetes Science and Technology, Nov. 2007
Block, Alexander, “Secret Sharing and 1-11 Threshold Signatures with BLS”, Jul. 2, 2018, https://blog.dash.org/secret-sharing-and-threshold-signatures-with-
Braun, “Infusomat® Space and Accessories”, Instructions for Use, Nov. 2010, pp. 68. <http://corp.bbraun.ee/Extranet/Infusionipumbad/Kasutusjuhendid/Vana
Brownlee, Seth, “Product Spotlight: The Plum A+ with Hospira MedNet Infusion System”, PP&P Magazine, Dec. 2005, vol. 2, No. 7, pp. 2. cited by applican
Cannon, MD et al., “Automated Heparin-Delivery System to Control Activated Partial Thromboplastin Time”, Circulation, Feb. 16, 1999, vol. 99, pp. 751-756
Cardinal Health, “Alaris® Syringe Pumps” Technical Service Manual, Copyright 2002-2006, Issue 9, pp. 1-88, <http://www.frankshospitalworkshop.com/equ
_Service_Manual.pdf>. cited by applicant
“CareAware® Infusion Management”, Cerner Store, as printed May 12, 2011, pp. 3, <https://store.cerner.com/items/7>. cited by applicant
Chen et al., “Enabling Location-Based Services on Wireless LANs”, The 11th IEEE International Conference on Networks, ICON 2003, Sep. 28-Oct. 1, 2003
“Computer Dictionary”, Microsoft Press, Third Edition, Microsoft Press, 1997, pp. 430 & 506. cited by applicant
“Context-Free Grammar”, Wikipedia.org, as last modified Mar. 5, 2010 in 11 pages, <https://en.wikipedia.org/w/index.php/?title=Context-free_grammar&old
Crawford, Anne J., MSN, RNC, “Building a Successful Quality Pain Service: Using Patient Satisfaction Data and the Clinical Practice Guideline”, USA, 1995
Crocker et al., “Augmented BNF for Syntax Specifications: ABNF”, Network Working Group, Standards Track, Jan. 2008, pp. 16. cited by applicant
Davidson et al., “A Computer-Directed Intravenous Insulin System Shown to be Safe, Simple, and Effective in 120,618 h of Operation”, Diabetes Care, Oct. 2
Davies, T., “Cordless Data Acquisition in a Hospital Environment”, IEE Colloquium on Cordless Computing—Systems and User Experience, 1993, pp. 4. cite
Dayhoff et al., “Medical Data Capture and Display: The Importance of Clinicians' Workstation Design”, AMIA, Inc., 1994, pp. 541-545. cited by applicant
Diabetes Close Up, Close Concerns AACE Inpatient Management Conference Report, Consensus Development Conference on Inpatient Diabetes and Metabo
Doesburg et al., “Improved Usability of a Multi-Infusion Setup Using a Centralized Control Interface: A Task-Based Usability Test”, Aug. 11, 2017, PLoS ON
“Download”, Free On-Line Dictionary of Computing, as archived Jun. 16, 2010 in 1 page, http://web.archive.org/web/20100616010314/https://foldoc.org/dow
East PhD et al., “Digital Electronic Communication Between ICU Ventilators and Computers and Printers”, Respiratory Care, Sep. 1992, vol. 37, No. 9, pp. 1
Edworthy, Judy, “Medical Audible Alarms: A Review”, Journal of the American Medical Informatics Association, vol. 20, No. 3, 2013, pp. 584-589. cited by a
Einhorn, George W., “Total Quality Pain Management: A Computerized Quality Assessment Tool for Postoperative Pain Management”, Abbott Laboratories, C
Eskew et al., “Using Innovative Technologies to Set New Safety Standards for the Infusion of Intravenous Medications”, Hospital Pharmacy, 2002, vol. 37, N
Felleiter et al., “Data Processing in Prehospital Emergency Medicine”, International journal of Clinical Monitoring and Computing, Feb. 1995, vol. 12, No. 1,
“File Verification”, Wikipedia.org, as last modified Oct. 11, 2011 in 2 pages, <https://en.wikipedia.org/w/index.php?title=File_verification&oldid=455048290
Fogt et al., Development and Evaluation of a Glucose Analyzer for a Glucose-Controlled Insulin Infusion System (Biostator®), Clinical Chemistry, 1978, vol.
Gabel et al., “Camp: A Common API for Measuring Performance”, 21st Large Installations System Administration Conference (LISA '07), 2007, pp. 49-61. ci
Gage et al., “Automated Anesthesia Surgery Medical Record System”, International Journal of Clinical Monitoring and Computing, Dec. 1990, vol. 7, No. 4,
Galt et al., “Personal Digital Assistant-Based Drug Information Sources: Potential to Improve Medication Safety”, Journal of Medical Library Association, Ap
Gardner, Ph.D et al., “Real Time Data Acquisition: Recommendations for the Medical Information Bus (MIB)”, 1992, pp. 813-817. cited by applicant
“General-Purpose Infusion Pumps”, Health Devices, EXRI Institute, Oct. 1, 2002, vol. 31, No. 10, pp. 353-387. cited by applicant
Givens et al., “Exploring the Internal State of User Interfaces by Combining Computer Vision Techniques with Grammatical Inference”, Proceedings of the 20
26, 2013, pp. 1165-1168. cited by applicant
Glaeser, “A Hierarchical Minicomputer System for Continuous Post-Surgical Monitoring”, Computers and Biomedical Research, Aug. 31, 1975, pp. 336-361.
Goldberg et al., “Clinical Results of an Updated Insulin Infusion Protocol in Critically Ill Patients”, Diabetes Spectrum, 2005, vol. 18, No. 3, pp. 188-191. cite
Gomez et al., “CLAM: Connection-Less, Lightweight, and Multiway Communication Support for Distributed Computing”, Computer Science, 1997, vol. 119
“GPS Tracker for Medical Equipment”, <http://www.trackingsystem.com/forbusinesses/corporate-trackingsystem/1098-gps-tracker-formedicalequipment.htm
Graseby, “Model 3000/500 and Micro 3100/505: Volumetric Infusion Pump”, Technical Service Manual, Graseby Medical Ltd., Apr. 2002, Issue A, pp. 160. c
Graseby, “Model 3000/500 and Micro 3100/505: Volumetric Infusion Pump: Illustrated Parts List for Pump Serial Nos. from 3000 to 59,999”, Technical Servi
Gutwin et al., “Gone But Not Forgotten: Designing for Disconnection in Synchronous Groupware”, CSCW 2010, Feb. 6-10, 2010, Savannah, Georgia, USA.,
Halpern et al., “Changes in Critical Care Beds and Occupancy in the United States 1985-2000: Differences Attributable to Hospital Size”, Critical Care Medic
Hamann et al., “PUMPSIM: A Software Package for Simulating Computer-Controlled Drug Infusion Pumps”, Annual International Conference of the IEEE E
cited by applicant
Hasegawa et al., “On a Portable Memory Device for Physical Activities and Informations of Maternal Perception”, Journal of Perinatal Medicine, 1988, vol. 1
Hawley et al., “Clinical Implementation of an Automated Medical Information Bus in an Intensive Care Unit”, Proceedings of the Annual Symposium on Com
Hayes-Roth et al., “Guardian: A Prototype Intelligent Agent for Intensive-Care Monitoring”, Artificial Intelligence in Medicine, vol. 4, Dec. 31, 1992, pp. 165
Hospira, GemStar® Pain Management Infusion System 9-084-PR1-2-2, <www.hospira.com/products/gemstar_painmanagement.aspx>, Jan. 28, 2010, pp. 1-2
Huang et al., “Secure Identity-Based Data Sharing and Profile Matching for Mobile Healthcare Social Networks in Cloud Computing”, vol. 6, Jul. 2018, pp. 3
Introducing Abbott TQPM (Total Quality Pain Management), Abbott Laboratories, Abbott Park, IL, May 2000, pp. 1-4. cited by applicant
“Infusion Pump”, Wikipedia.org, as last modified Mar. 27, 2014, in 3 pages, <https://web.archive.org/web/20140703024932/https://en.wikipedia.org/wiki/Infu
Isaka et al., “Control Strategies for Arterial Blood Pressure Regulation”, IEEE Transactions on Biomedical Engineering, Apr. 1993, vol. 40, No. 4, pp. 353-36
Johnson et al., “Using BCMA Software to Improve Patient Safety in Veterans Administration Medical Centers”, Journal of Healthcare Information Manageme
Kent Displays, “Reflex™ Electronic Skins”, Product Brief 25127B, 2009, pp. 2. cited by applicant
Kent Displays, “Reflex Electronic Skins Engineering Evaluation Kit”, 25136A, Mar. 10, 2009. cited by applicant
Lefkowitz et al., “A Trial of the Use of Bar Code Technology to Restructure a Drug Distribution and Administration System”, Hospital Pharmacy, Mar. 31, 19
Lenssen et al., “Bright Color Electronic Paper Technology and Applications”, IDS '09 Publication EP1-2 (Phillips Research), 2009, pp. 529-532. cited by appl
Leveson, Nancy, “Medical Devices: The Therac-25”, Appendix A, University of Washington, 1995, pp. 49. cited by applicant
Li et al., “Hijacking an Insulin Pump: Security Attacks and Defenses for a Diabetes Therapy System”, 2011 IEEE 13th International Conference on e-Health N
Linkens, D.A. “Computer Control for Patient Care”, Computer Control of Real-Time Processes, IEE Control Engineering Series 41, 1990, Ch. 13, pp. 216-23
Mako Hill et al., “The Official Ubuntu Book”, Shoeisha Co., Ltd., 1st Edition, Jun. 11, 2007, pp. 115 to 125. cited by applicant
Marshall, et al., “New Microprocessor-Based Insulin Controller”, IEEE Transactions on Biomedical Engineering, Nov. 1983, vol. BME-30, No. 11, pp. 689-69
Martino et al., “Automation of a Medical Intensive Care Environment with a Flexible Configuration of Computer Systems”, Proceedings of the Annual Sympo
cited by applicant
Matsunaga et al., “On the Use of Machine Learning to Predict the Time and Resources Consumed by Applications”, 2010 10th IEEE/ACM International Conf
504. cited by applicant

Mauseth et al., “Proposed Clinical Application for Tuning Fuzzy Logic Controller of Artificial Pancreas Utilizing a Personalization Factor”, Journal of Diabet
“McKesson Automation and ALARIS Medical Systems Developing Point-of-Care Bar Coding Solution to Improve IV Medication Safety”, PR Newswire, NY
Medfusion™, “Medfusion Syringe Infusion Pump Model 4000”, Operator's Manual, Software Version V1.1, Sep. 2011, pp. 154. <http://www.medfusionpump
applicant
Metnitz et al., “Computer Assisted Data Analysis in Intensive Care: the ICDEV Project-Development of aScientific Database System for Intensive Care”, Inte
pp. 147-159. cited by applicant
Michienzi, Kelly, “Managing Drug Library Updates”, Pharmacy Purchasing Products, https://www.pppmag.com/article/1061, Feb. 2012, vol. 9, pp. 22-23. cit
Micrel Medical Devices, “MP Daily +” <http://web.archive.org/web/20130803235715/http://www.micrelmed.com/index.aspx?productid=9> as archived Aug.
Moghissi, Etie, MD, FACP, FACE, “Hyperglycemia in Hospitalized Patients”, A Supplement to ACP Hospitalist, Jun. 15, 2008, pp. 32. cited by applicant
Murphy, Robert, “The Design of Safety-Critical Medical Infusion Devices”, May 30, 2007, Doctor of Philosophy submission, pp. 317. cited by applicant
Murray, Jr. et al., “Automated Drug Identification System (during surgery)”, IEEE Proceedings of Southeastcon '91, Apr. 7-10, 1991, pp. 265. cited by applica
Nicholson et al., “Smart' Infusion Apparatus for Computation and Automated Delivery of Loading, Tapering, and Maintenance Infusion Regimens of Lidocain
on Computer Applications in Medical Care, Oct. 1983, pp. 212-213. cited by applicant
Nojoumian et al., “Social Secret Sharing in Cloud Computing Using a New Trust Function”, 2012 Tenth Annual International Conference on Privacy, Security
Nolan et al., “The P1073 Medical Information Bus Standard: Overview and Benefits for Clinical Users”, 1990, pp. 216-219. cited by applicant
Omnilink Systems, Inc., “Portable Medical Equipment Tracking”, <http://www.omnilink.com/portablemedicalequipmenttracking/>, Mar. 15, 2015, pp. 2. cite
O'Shea, Kristen L., “Infusion Management: Working Smarter, Not Harder”, Hospital Pharmacy, Apr. 2013, vol. 48, No. 3, pp. S1-S14. cited by applicant
Package Management in Debian GNU/Linux, Debian GNU/Linux Expert Desktop Use Special, Giutsu-Hyohron Co., Ltd., First Edition, Sep. 25, 2004, pp. 18
Passos et al., “Distributed Software Platform for Automation and Control of General Anaesthesia”, Eighth International Symposium on Parallel and Distribute
Philips, “IntelliSpace Event Management and IntelliVue Patient Monitoring”, Release 10, 2011,
<http://incenter.medical.philips.com/doclib/enc/fetch/2000/4504/577242/577243/577247/582646/583147/8359175/Philips_Patient_Monitoring_and_IntelliSp
2>, pp. 2. cited by applicant
Pretty et al., “Hypoglycemia Detection in Critical Care Using Continuous Glucose Monitors: An in SilicoProof of Concept Analysis”, Journal of Diabetes Scie
Rahmani et al., “Smart e-Health Gateway: Bringing Intelligence to Internet-of-Things Based Ubiquitous Healthcare Systems”, 2015 12th Annual IEEE Consu
cited by applicant
Rappoport, Arthur E., “A Hospital Patient and Laboratory machine-Readable Identification System (MRIS) Revisited”, Journal of Medical Systems, Apr. 1984
Ritchie et al., “A Microcomputer Based Controller for Neuromuscular Block During Surgery”, Annals of Biomedical Engineering, Jan. 1985, vol. 13, No. 1, p
Saager et al., “Computer-Guided Versus Standard Protocol for Insulin Administration in Diabetic Patients Undergoing Cardiac Surgery”, Annual Meeting of th
applicant
Sanders et al., “The Computer in a Programmable Implantable Medication System (PIMS)”, Proceedings of the Annual Symposium on Computer Application
Schilling et al., “Optimizing Outcomes! Error Prevention and Evidence-Based Practice with IV Medications”, A Pro-Ce Publication, Hospira, Inc., Feb. 6, 201
Schulze et al., “Advanced Sensors Technology Survey”, Final Report, Feb. 10, 1992, pp. 161. cited by applicant
Scott, et al., “Using Bar-Code Technology to Capture Clinical Intervention Data in a Hospital with a Stand- Alone Pharmacy Computer System”, Mar. 15, 199
applicant
Sebald et al., “Numerical Analysis of a Comprehensive in Silico Subcutaneous Insulin Absorption Compartmental Model”, 31st Annual International Conferen
3901-3904. cited by applicant
Shabot, M. Michael, “Standardized Acquisition of Bedside Data: The IEEE P1073 Medical Information Bus”, International Journal of Clinical Monitoring and
Sheppard, Louis, Ph.D., “Automation of the Infusion of Drugs Using Feedback Control”, Journal of Cardiothoracic and Vascular Anesthesia, Feb. 28, 1989, v
Sheppard, Louis, Ph.D., “Computer Control of the Infusion of Vasoactive Drugs”, Annals of Biomedical Engineering, Jul. 1980, vol. 8, No. 4-6, pp. 431-444.
Sheppard, Louis, Ph.D., “The Application of Computers to the Measurement, Analysis, and Treatment ofPatients Following Cardiac Surgical Procedures”, The
applicant
Sheppard, Louis, Ph.D., “The Computer in the Care of Critically Ill Patients”, Proceedings of the IEEE, Sep. 1979, vol. 67, No. 9, pp. 1300-1306. cited by app
“Sigma Spectrum: Operator's Manual”, May 15, 2008, pp. 63. <https://usme.com/content/manuals/sigma-spectrum-operator-manual.pdf>. cited by applicant
“Sigma Spectrum: Operator's Manual”, Oct. 2009, pp. 72. <http://static.medonecapital.com/manuals/userManuals/Sigma-Spectrum-Operator-Manual-October
Simonsen, Michael Ph.D., POC Testing, New Monitoring Strategies on Fast Growth Paths in European Healthcare Arenas, Biomedical Business & Technolog
Siv-Lee et al., “Implementation of Wireless ‘Intelligent’ Pump IV Infusion Technology in a Not-for-Profit Academic Hospital Setting”, Hospital Pharmacy, Se
cited by applicant
Slack, W.V., “Information Technologies for Transforming Health Care”, <https://www.andrew.cmu.edu/course/90-853/medis.dir/otadocs.dir/03ch2.pdf>, Ch. 2
Smith, Joe, “Infusion Pump Informatics”, CatalyzeCare: Transforming Healthcare, as printed May 12, 2011, pp. 2. cited by applicant
Sodders, Lisa, “VA Center Keeps Medicine in Right Hands”, The Capital-Journal, Dec. 4, 1999, pp. 1-2. cited by applicant
“Software Versioning”, Wikipedia.org, dated Oct. 16, 2011 in 11 pages, <https://en.wikipedia.org/w/index.php?title=Software_versioning&oldid=455859110>
Solapurkar et al., “Building Secure Healthcare Services Using OAuth 2.0 and JSON Web Token in IOT Cloud Scenario”, Dec. 2016, 2nd International Confer
Stitt, F.W., “The Problem-Oriented Medical Synopsis: a Patient-Centered Clinical Information System”, Proceedings of the Annual Symposium on Computer
Stokowski, Laura A. RN, MS, “Using Technology to Improve Medication Safety in the Newborn Intensive Care Unit”, Advances in Neonatal Care, Dec. 2001
Sutton et al., “The Syntax and Semantics of the PROforma Guideline Modeling Language”, Journal of the American Medical Informatics Association, Sep./O
Szeinbach et al., “Automated Dispensing Technologies: Effect on Managed Care”, Journal of Managed Care Pharmacy (JMCP), Sep./Oct. 1995, vol. 1, No. 2,
Szolovits et al., “Guardian Angel: Patient-Centered Health Information Systems”, Technical Report MIT/LCS/TR-604, Massachusetts Institute of Technology
“TCG TPM v2.0 Provisioning Guidance”, Reference, Version 1, Revision 1, Mar. 15, 2017, pp. 1-43. cited by applicant
Van Den Berghe, M.D., Ph.D., et al., “Intensive Insulin Therapy in Critically Ill Patients”, The New England Journal of Medicine, Nov. 8, 2001, vol. 345, No.
Van Den Berghe, M.D., Ph.D., et al., “Intensive Insulin Therapy in the Medical ICU”, The New England Journal of Medicine, Feb. 2, 2006, vol. 354, No. 5, pp
Van Der Maas et al., “Requirements for Medical Modeling Languages”, Journal of the American Medical Informatics Association, Mar./Apr. 2001, vol. 8, No.
Villalobos et al., “Computerized System in Intensive Care medicine”, Medical Informatics, vol. 11, No. 3, 1986, pp. 269-275. cited by applicant
Wilkins et al., “A Regular Language: The Annotated Case Report Form”, PPD Inc., PharmaSUG2011—Paper CD18, 2011, pp. 1-9. cited by applicant
Ying et al., “Regulating Mean Arterial Pressure in Postsurgical Cardiac Patients. A Fuzzy Logic System to Control Administration of Sodium Nitroprusside”,
pp. 671-677. cited by applicant
Yoo et al., “Code-Based Authentication Scheme for Lightweight Integrity Checking of Smart Vehicles”, IEEE Access, 2018, vol. 6, pp. 46731-46741. cited by
Yue, Ying Kwan, “A Healthcare Failure Mode and Effect Analysis on the Safety of Secondary Infusions”, Thesis, Institute of Biomaterials and Biomedical En
Yurkonis et al., “Computer Simulation of Adaptive Drug Infusion”, IEEE Transactions on Biomedical Engineering, vol. BME-34, No. 8, Aug. 1987, pp. 633-6
Zakariah et al., “Combination of Biphasic Transmittance Waveform with Blood Procalcitonin Levels for Diagnosis of Sepsis in Acutely Ill Patients”, Critical C
International Search Report and Written Opinion received in PCT Application No. PCT/US2014/021335, dated Jul. 8, 2014 in 13 pages. cited by applicant
International Preliminary Report on Patentability and Written Opinion received in PCT Application No. PCT/US2014/021335, dated Sep. 17, 2015 in 12 page

---

Primary Examiner: Onamuti; Gbemileke J

Attorney, Agent or Firm: Knobbe, Martens, Olson & Bear, LLP

Background/Summary

INCORPORATION BY REFERENCE TO ANY PRIORITY APPLICATIONS

(1) Any and all applications for which a foreign or domestic priority claim is identified in the Application Data Sheet as filed with the present application are hereby incorporated by reference under 37 CFR 1.57.

BACKGROUND OF THE INVENTION

Field of the Invention

(2) One or more embodiments of the invention are related to the field of multiplex communication protocols for medical devices such as, but not limited to, infusion pumps. More particularly, but not by way of limitation, embodiments of the invention enable a medical device communication method for communication between connected peripherals and subsystems that includes connection-oriented, connectionless-oriented, broadcast and multicast data exchange with priority handling of data, fragmentation and reassembly of data, unique static and dynamic address assignment and hot swap capabilities.

Description of the Related Art

(3) Devices that exchange data generally do so using a communication protocol. Communication protocols enable data to be transmitted and received in a controlled manner. Medical devices are example devices that may utilize a communication protocol, for example to exchange data between peripherals or subsystems that generate or utilize data. There are many types of communications protocols that vary in complexity, efficiency and hardware utilization. Current communication protocols utilized within medical devices make use of the operating system and particular bus architecture within the medical device. A problem with this type of architecture is that some implementations may prevent time-multiplexed access of the communication link, thereby starving or otherwise preventing multiple applications from communicating simultaneously. In addition, applications that transfer data using operating system and bus specific software calls must be altered when the operating system or bus architecture changes, specifically to account for differences in operating system calls or with respect to the bus architecture, different data formatting, sequencing and any other protocol specific nuances. In addition, medical devices in general must undergo extensive testing to ensure that they do not fail. Thus, changing bus architectures increases costs associated with applications that make use of the bus architecture, since the application must be retested if the source code for the application is altered.

(4) Known communications protocols are generally targeted at a specific type of communication bus architecture, for example Ethernet, WiFi, Bluetooth, CAN, Serial, I2C, SPI, etc. Known communication protocols in general are not capable of use with more than one type of communication bus since they attempt to provide a solution to a specific communication problem in a coherent manner. Because of the low power requirements, limited processor capabilities and limited memory capacity of medical devices with embedded processors that do specific functions or tasks, such as infusion pumps, existing sophisticated communications protocols are generally not utilized in such medical devices.

(5) In summary, known solutions use communication protocols that are tied to a specific operating system and/or communications bus. Unfortunately, these communication protocols are not agnostic to all communication bus types and do not provide an efficient and lightweight protocol stack for intra-device communication that includes connection-oriented, connectionless-oriented, broadcast and multicast data exchange with priority handling of data, fragmentation, and reassembly of data, unique static and dynamic address assignment for connected subsystems and hot swap capabilities. For at least the limitations described above there is a need for a medical device communication method that provides these features as described and claimed herein.

SUMMARY OF THE INVENTION

(6) Embodiments of the invention enable a medical device communication method for communication between medical peripherals and subsystems that includes connection-oriented, connectionless-oriented, broadcast and multicast data exchange with priority handling of data, fragmentation and reassembly of data, unique static and dynamic address assignment and hot swap capabilities. Example medical devices that may employ an embodiment of the invention include but are not limited to infusion pumps, both present and future. Embodiments of the communication protocol provide an interface that is detached, or otherwise abstracted from the operating system and underlying bus architecture within the medical device, making the behavior and interface of communication protocol consistent across bus architectures and operating systems, which is unknown in the art of infusion pumps for example. Hence, the same application may be utilized on multiple hardware platforms, for example without altering the application itself. Thus, embodiments enable simplified application code, portability thereof and minimize maintenance and testing requirements. Embodiments may utilize any type of physical communication path, for example wireless or hardwired, including but not limited to a data bus. Embodiments for intra-device communications over a data bus generally employ data bus drivers specific to each type of data bus to control reading and writing of data over the bus along with a standard interface to these data bus drivers.

(7) Embodiments may be implemented in separate layers of software configured to execute on one or more computing elements, wherein each layer performs operations to provide data exchange that is generally independent of the other layers. Each layer for example may create, read or update headers associated with data to be exchanged, wherein the headers contain information to support the above-mentioned features. The layers make up what is known as a protocol stack. Embodiments of the protocol stack may include a manager layer, session layer, transport layer, and data link layer or any other architecture as long as the resulting implementation provides the functionality described herein.

(8) Depending on the peripheral or subsystem, data type, priority and desired reliability of data to be exchanged, applications may transmit data using connection-oriented data exchange to provide guaranteed delivery of data or connectionless data exchange for less sensitive data. Embodiments also support one-to-one, as well as one-to-many and many-to-one multicast, and broadcast modes of data exchange between connected peripherals and subsystems. At least one embodiment also supports priority based data exchange and gives preference to high priority data over low priority data to ensure that high priority messages are delivered first. Additionally, at least one embodiment supports data fragmentation and reassembly data to comply with demands of the particular physical communication technology. Embodiments also provide unique static and dynamic address assignment for connected subsystems and hot swap capabilities, which are unknown for example in current infusion pumps.

(9) Specifically, in the case of connection-oriented communication, at least one embodiment utilizes a Communication ID or “CID”, as a token to uniquely identify all active connections within a subsystem and route the data between respective applications. In the case of connectionless communications, at least one embodiment uses port numbers, for example source and destination port numbers, to identify the targeted application. At least one embodiment supports subscription services for recipient applications, which enables multicasting of data to all subscribed applications. Multicasting can be both connection-oriented and connectionless. In connection-oriented communication sessions, at least one embodiment guarantees delivery of data, for example using acknowledgements. Alternatively, connectionless communication sessions do not guarantee delivery of data, but are very efficient. At least one embodiment supports broadcasting of data/messages, wherein the broadcast messages are forwarded to all the subsystems connected to the broadcasting subsystem.

(10) Applications may need to exchange data larger in size than an underlying communication technology or data bus can support. In such cases, at least one embodiment breaks or fragments the data into a smaller size, for example that the data bus can actually transfer. At least one embodiment reassembles data into the original data size at the receiving end. At least one embodiment executes on embedded systems that may have limited resources, including memory, processing power, bus utilization, and power. Hence, embodiments efficiently utilize available resources. Example data

exchanges that are large enough to warrant fragmentation of messages include drug library downloads and firmware updates.
(11) With respect to fragmentation, at least one embodiment utilizes window that represents a count of fragments that may be sent before receiving an acknowledgement from receiver. In at least one embodiment, the transmitter requests for window size from the receiver before sending the first fragment. The receiver determines the available memory space to accommodate received packets and responds with the window size, for example as an integral multiple of the maximum frame size that fits into the available memory. The transmitter numbers the fragments in sequence and sends them to receiver. After a window size worth of messages have been sent, the transmitter waits for an acknowledgement of the last fragment. The receiver accumulates all the received fragments and verifies that all the received fragments are in sequence. If there is no missing fragment, the receiver sends the fragment number of last fragment as an acknowledgement, or otherwise sends the fragment numbers of missing fragments as part of negative acknowledgement or NAK.
(12) Since medical devices such as infusion pumps in the future may include hot swappable peripherals or subsystems, at least one embodiment supports unique address assignments to connected devices in order to provide conflict free exchange of data, thus reducing complexity in applications. At least one embodiment supports communication over multiple underlying data transfer technologies such as serial, CAN, SPI, SDIO, USB, or any other type of physical medium or data bus. At least one embodiment also keeps track of devices connected on each bus and routes data onto the respective bus.

Description

BRIEF DESCRIPTION OF THE DRAWINGS
(1) The above and other aspects, features and advantages of the invention will be more apparent from the following more particular description thereof, presented in conjunction with the following drawings wherein:
(2) FIG. **1** illustrates an architectural view of a system having a user interface controller, and multiple peripherals that communicate with one another using an embodiment of the invention.
(3) FIG. **2** illustrates a hierarchical layered embodiment of the invention implemented as a protocol stack.
(4) FIG. **3** illustrates an embodiment of an address request method implemented within the manager layer.
(5) FIG. **4** illustrates an embodiment of a simple infusion sequence utilizing various messages provided by embodiments of the method.
(6) FIG. **5** illustrates an embodiment of a connection method implemented within the session layer.
(7) FIG. **6** illustrates an embodiment of a data exchange method implemented within the session layer.
(8) FIG. **7** illustrates an embodiment of a disconnection request method implemented within the session layer.
(9) FIG. **8** illustrates a layer flow diagram that shows the flow of data within the various layers implemented in at least one embodiment of the invention.
(10) FIG. **9** illustrates an activity diagram showing routing between various devices.
(11) FIGS. **10**A-D illustrate the structure of the messages of the Session Layer.
(12) FIGS. **11**A-B illustrate the structure of the messages of the Transport Layer.
(13) FIGS. **12**A-B illustrate the structure of the messages of the Data Link/Physical Layer.
(14) FIGS. **13**A-B illustrate an exemplary message transfer of a medical function using exemplary values within the messages to demonstrate the system and method according to at least one embodiment of the invention.
DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT
(15) A medical device communication method will now be described. In the following exemplary description numerous specific details are set forth in order to provide a more thorough understanding of embodiments of the invention. It will be apparent, however, to an artisan of ordinary skill that the present invention may be practiced without incorporating all aspects of the specific details described herein. In other instances, specific features, quantities, or measurements well known to those of ordinary skill in the art have not been described in detail so as not to obscure the invention. Readers should note that although examples of the invention are set forth herein, the claims, and the full scope of any equivalents, are what define the metes and bounds of the invention.
(16) FIG. **1** illustrates an architectural view of a system having user interface controller or “UIC”, and multiple peripherals that communicate with one another using an embodiment of the invention. As shown user interface controller UIC communicates with peripherals Pump Motor Control or “PMC”, PMC 1 and PMC 2 as well as communication engine or “CE” for various applications including but not limited to drug library, status and diagnostic message handling. For exemplary purposes, UIC has a destination/device ID, e.g., an address of 5 and messages from UIC to the other devices travel over pathways uniquely defined by the tuples defined in the table, for example on a per device and communication ID defined channel. These channels are shown in the table above UIC, namely between UIC and PMC 1, at ports 10 and 20, i.e., the therapeutic and status ports, via Communication ID or “CID” **100** and CID **250** respectively followed by a channel used between UIC and PMC 2 at port 10, the therapeutic port, via Communication ID **100**, along with a channel between UIC and CE at port 40, via Communication ID **175**. The CE, whose address is 7, shows channels in the table above CE to PMC 2 and the UIC, namely devices 2 and 5 via Communication ID's **250** and **175** respectively. PMC 1 is illustrated as having channels to the UIC, via Communication ID's **100** and **250**. PMC 2 is illustrated as having channels to the UIC and CE through ports 10 and 20, via Communication ID's **100** and **250**. PMC 3 and 4 may be hot swapped into the system or otherwise commanded or queried on the fly. Embodiments of the invention are generally configured to utilize minimal memory and processing to enable execution on devices having limited memory and limited processing power, which is generally unknown in the art with respect to sophisticated communications protocols for example. In one or more embodiments, the stack utilizes one kernel thread to execute the Data Link layer and Transport lower layer, whereas remaining layers are part of application process and execute in the context of application. Minimum thread implementation supports blocking access, for example read and write operations block until the operation is completed. Embodiments may also support asynchronous callbacks, and in such cases, the stack may utilize two threads, one for write operations and one for read operation, hence total number of threads utilized is 2*N+1, where N is the number of applications using the stack.
(17) FIG. **2** illustrates a hierarchical layered embodiment of the invention implemented as a protocol stack. As shown, a data message in the application layer is N bytes long. The application layer may include any functionality independent of the protocol stack that is implemented in the layers beneath the application layer as shown. When the message is transmitted from one application to another, for example to an application executing on a peripheral or subsystem, control information or headers are appended to the message as the message descends layers. The various headers or other appended information are removed as the message rises through the protocol stack to the respective receiving application.
(18) In one or more embodiments, a manager layer may be utilized to implement the first layer in the protocol stack beneath the application. The manager layer may provide standard interfaces to applications across any desired operating system. The layer provides application programmer interfaces or API's that enables socket-based communications between applications. The manager layer also manages file descriptors and facilitates opening of ports. In at least one embodiment, the manager layer creates and otherwise utilizes a message header having a port number and file descriptor.
(19) A session layer is another layer in the protocol stack and provides or includes API's to exchange data and control between manager layer and session layer. The session layer may provide guaranteed application-to-application delivery of data and enables connection-oriented and connectionless-oriented modes of communication. This layer also enables one-to-one, one-to-many, many-to-one multicast and broadcasting mode of communication. The layer maintains the translation between CID and an associated socket or virtual port. For connection-oriented communication,

the protocol utilizes the CID and otherwise generates and utilizes CID's. As the connection-oriented data exchange utilizes a handshake between applications for data exchange, the session layer handles the handshake and generates a CID for the communication and informs the other participating session layers of application(s) about the CID. After the handshake, data packets utilize the CID for communication. In case of connectionless communication, no CID is utilized and hence both source and destination port addresses are exchanged in each communication packet or payload. In at least one embodiment, the manager layer creates and otherwise utilizes a message header having control flags and a message type along with a communication identifier. This structure along with an exemplary connection table is shown in FIG. **10**A, along with exemplary message types in FIGS. **10**B-D. The control flags may be implemented with a layer flag of 1 bit, a connection type of 2 bits and a CID source of 1 bit for example. The session layer utilizes some messages that are associated with the session-session communications and are never passed up the stack to the manager layer in one or more embodiments. These messages are generally used for establishing or closing connections, acknowledgements, etc. If the layer flag is set, for example set to True or 1, the message will be consumed at session layer and will not be forwarded up the stack. The connection type flag indicates the type of connection, for example if connection-oriented, set to 01 or if connectionless, set to 00. An example connectionless protocol is User Datagram Protocol or UDP while an example connection-oriented protocol is Transmission Control Protocol or TCP. The CID source bit is used to identify if the data as being sent from the entity that generated CID for the connection in use or from sub-modules using this CID for communication. The entity that generates CID for communication sets this bit for all the messages generated by it for the respective active connection, while other entities involved in communication reset this flag for messages while using this CID. As the CID is unique within the entity generating CID, there may be duplicate CIDs across other entities. Hence, this layer helps in resolving the source of CID (local or remote) via this flag. The message type field associates messages with categories and lets the session layer know what to expect in the following fields. The message type field is a 4-bit wide field in one or more embodiments. The message type field is used to determine the type of message. Exemplary values include 0000 for data, 0001 for connection, 0010 for CID, 0011 for socket, 0100 for service and 0101 for device information. Any module that provides a service generates a unique CID for communicating with the consumers of the service. Communication ID ‘0’ is reserved for connectionless type of communication in one or more embodiments. Communication ID field is 1 byte wide and is utilized for the data that is passed up the protocol stack. CID can hold any number between 0-255. As state above, CID ‘0’ is for connectionless type communication and is thus not a valid ID for connection-oriented communication. Connection oriented type communications will have a CID in the range of 1-255. Hence, CID ‘0’ is an implicit indication of connectionless communication, any other number between 1-255 suggests connection-oriented. Applications may establish one or more notification filters to select message to receive and process using a desired function. The filtration mechanism may utilize one or more regular expression that specifies the location, length and content of the matching data in the data portion of the packet. This functionality is implemented in the management layer in one embodiment of the invention. Before the management layer forwards the data to application, it may check if any filters are defined on the data. Depending on the filter, the manager layer filters data and forwards the data to respective callback handlers.

(20) Embodiments of the invention enable a single application to maintain connections with more than one device over one or more physical communication layers or bus implementations. This is accomplished by the use of virtual ports. A single application such as the Therapeutic Manager in the UIC may for example maintain open connections with more than one drug pump PMC or other device as would be asserted during a multi-channel infusion. Similarly, many applications may maintain a connection with one application or device, for example, UIC, CE, and other applications may connect to a particular PMC to gather infusion status information.

(21) The one-to-many and/or many-to-one communication relationship can further be classified into three types, unicast, multicast and broadcast. For example, different applications can gather infusion status from a PMC either by requesting, for example via multicasting, or the PMC can broadcast its status on a known port and interested applications can listen to the port. Listening to a known port can be either anonymous or subscription based. In anonymous mode, broadcasting application continuously transmits on a known port and any application can listen to the port. In subscription based mode, the broadcasting application will not transmit until there is at least one recipient, interested application, which will have to request for service and disconnect when done using the service.

(22) Virtual ports can be implemented by enabling a handshake between participating modules/applications. Applications providing the service generally open a port and connect to the port. For every accepted connection request, CID is generated by the service provider and is passed back to requesting entity in an acknowledgement. Subsequent communication is performed using this CID. In general, the CID is unique to the entity that generated it. A disconnection message is used to stop communication and the CID is then returned to the pool, for example to be reused later. If the service provider runs out of CIDs, it may return a NAK to incoming connection requests with appropriate NAK ID. In case of communication failure, for example module shut down, too much waiting time, too many retries, etc., after waiting for sufficient retries to send a message, one or more embodiments may assume that the communication has stopped and CID is then returned to pool. As the CID are generated by the service provider and are unique within the entity, there can be duplicate CIDs on other sub-entities. To avoid the conflict because of duplicate CIDs, two CID tables may be maintained, one for the CID generated by the system, and the other for the CIDs generated by other systems engaged in communication. The creator of CID sets the “CID Source” flag, hence when other involved applications look at this flag, they perform lookup in appropriate CID table. Each entity may therefor maintain a table shared by the applications running on it. This table is maintained at the session layer and serves as a reference table for routing incoming data packets to respective ports/sockets.

(23) As example scenario is illustrated in the following table, and is also shown in the bottom portion of FIG. **10**A for illustration purposes and is not intended to limit the invention as claimed. As shown, the connection type may be set to a value indicative of a connection-oriented type of communication, such as TCP as shown, or a connectionless communication type, such as UDP as shown, or a “Service”, for example an application that exists to log data for other applications. The destination address, destination port and communication ID generally uniquely identify a row in above-mentioned table. Destination address is the logical address of a device engaged in a communication. Embodiments may support repeated entries with the same destination address, which indicates multiple active connections with the same destination device. The source port field stores the local port number responsible for handling communication(s) with the CID associated therewith. Depending on CID, received messages are routed to the respective port. Multiple repeated entries in the source port column suggest various applications communicating over same port, which may be indicative of one-to-many communication for example. In one or more embodiments, applications may register or otherwise provide a request to a service provider to receive messages. The destination port is the port number on the destination device engaged in a communication. The communication between a destination port and the local port associated therewith takes place over the respective CID. Hence, CID behaves as a key for this communication. Since the CID is a unique number assigned to distinct communication requests, and which may be implemented with a particular data type of a certain size, there may be an upper limit to the number of active connections that can be handled by the system/application. The upper limit is thus an upper numerical limit of the CID. Once the count of unique CID's exceeds the upper limit, one or more embodiments send a NAK to new incoming connection requests. The File Descriptor (FD) functions similar to file handler or file descriptor in standard operating systems as one skilled in the art will recognize. Communication related operations are performed using this descriptor. Repeating entries of FD suggests multiple connections are being served by one application, many-to-one type of communication. See also FIGS. **10**B-D for specific message structures utilized in one or more embodiments of the invention.

(24) TABLE-US-00001 Connection Destination Destination Source File Type Address Port Port CID Descriptor Service 8 50 40 100 55 TCP 5 23 60 72 63 Service 15 68 40 110 87 UDP 4 20 55 103 21

(25) The transport layer is another layer in the protocol stack and is responsible for transport layer to transport layer delivery of data. This layer handles flow control, timeouts, acknowledgements and fragmentation and reassembly of data and also resolves the data priority. At least one embodiment of the protocol stack supports two or more priority levels, for example three priority levels, High priority, Medium priority and Low

priority and depending on the priority of data, the transport layer puts the data in a respective priority queue. The transport layer may be implemented with two sub-layers namely the transport upper and lower layers. The transport upper layer along with manager and session layers resides in application space, whereas the transport lower layer along with data link layer resides in kernel space. The transport upper layer handles reading and writing to priority queues, fragmentation and reassembly of data and transport-to-transport layer acknowledgements, whereas the transport lower layer may be implemented as a very thin layer and handles reading from priority queues and communication with one or more other stack layers, for example a lower stack layer. This structure along with an exemplary message types in FIGS. **11**A-B.

(26) The transport layer generally ensures that manageable sized datagrams are sent over the underlying bus. Hence, this layer looks at the data coming from upper layers and if the size of data exceeds Maximum Transmission Unit (MTU) size, the layer fragments the incoming data to fit within MTU boundary. Thus, embodiments of the invention may utilize any type of bus of any size, e.g., one bit as per a serial bus, or multiple bits as per a parallel bus of any width. The layer adds appropriate information to the data so that it can be reassembled faithfully at the receiving end. If the incoming data can be sent in three fragments, 'Fragment ID' field is used to number the fragments starting from '1' and the 'Extended flag' bit is not used. All zeros in the 'Fragment ID' field indicates an un-fragmented message and hence is treated as a standalone message. If a message requires more than three fragments to be transmitted, 'Extended Flag' is set, which enables an extra of 8 bits (Extended Fragment ID field is 8 bits) to be used for numbering the fragments. With this flag set, there are total of 10 bits available for numbering which can support 1023 (2{circumflex over ( )}10−1) fragments. At the receiving end, 'Extended flag' is inspected to determine if 'Extended Fragment ID' is used or not. If the flag is set, the receiver assumes the fragments to arrive in sequence, starting from sequence number 1. But, if the flag is not set, the receiver inspects the 'Fragment ID' field. If the 'Fragment ID' field has zero in it, it indicates an independent message, but if it's a non-zero value, the receiver treats the received message as fragmented data (expects a maximum of three packets). Once all of the fragments are received, the receiver will re-assemble all the fragments into one message. To do this, the receiver aligns all the received messages in ascending order of their fragment ID. Then the receiver verifies that no fragment has been missed in the sequence. If all fragments are received successfully, the receiver removes the 'Transport layer' header information from all the related fragments and concatenates them into one message. If Transport layer has limited memory to re-assemble all the fragments, it forwards the fragments up the stack, as they arrive, which gets reassembled in application buffer.

(27) Congestion control is also provided by the transport layer, which may implement messages dedicated specifically for transport layer to layer communication. These specific messages are consumed at transport layer and not passed up the stack. One such message is the window message, which is exchanged to determine window size for data exchange.

(28) Before sending the first fragment from fragmented data, the transmitter requests a window size from receiver. The receiver looks at the available buffer space in the application buffer and computes the number of fragments it can stage before running out of available memory. It responds to transmitters request with this computed number as window size. Then the transmitter sends window size worth of fragments before expecting an acknowledgement. Once the receiver receives all the messages transmitted in a window, it verifies that all the fragments are in desired sequence and sends acknowledgement for last received fragment in the sequence. If the receiver determines that fragment(s) is missing, it sends an NAK for the missing fragment and the transmitter re-transmits the respective fragment(s). The transmitter may check for window size in middle of communication to keep the data exchange optimized, also, if the receiver gets low on resources, it can explicitly send a window response and update the transmitter about the window size.

(29) The transport layer is also responsible for the reliable delivery of data. The transport layer has ability to ensure delivery of data to the receiving end. Transport layer has a field for acknowledgement. The receiver may send an acknowledgement for every received data packet with the acknowledgement flag set. In case of fragmented messages, an acknowledgement is sent when the last fragment in a window has been received or last frame in the message has been received or timer expires before all messages have been received.

(30) Embodiments of the transport layer may also implement a "time to live". For example, after transmitting a message, the transmitter initiates a timer and waits for an acknowledgement. If acknowledgement is received, the timer is reset and next packets are transmitted. But if no acknowledgement is received, the transport layer re-transmits the message and again waits for an acknowledgement. The transmitter will retry to send the message certain number of times and if it fails to get an acknowledgement, it will assume that the receiver is not available and will inform upper layers. In case of fragmentation, the transmitter sends window-sized messages and then waits for an acknowledgement on the last fragment sent. If the timer expires, the transmitter will resend the messages again.

(31) The transport layer also may implement fault detection and recovery. For example, the transport layer at the receiver may request the transmitter to re-transmit selected frames through layer-to-layer messages.

(32) The transport layer may also implement priority for messages. For example, the upper layers may pass the message priority down to this layer and this layer adds the priority to the message header. Header has a two bit fields for message priority and hence there are four priority levels possible in one or more embodiments although any number of bits may be used for priority to implement more levels and this applies to all message partitions and bit numbers described herein. Each priority level has its own queue and depending on message priority, transport layer puts them into respective queues to be processed by other layers. As there are four priority levels in a 2-bit embodiment, there may be a maximum of four priority queues and a minimum of one queue, but the number of priority queue depends on the number of priority levels used.

(33) The data link layer is another layer, by way of example and not limitation the bottommost layer, in the communication stack and is responsible for subsystem-to-subsystem delivery of data. This layer completely resides in the kernel space. Data link layer may also be implemented with two sub-layers, for example a Link Layer and Media Access (MAC) layer. The link layer verifies data integrity by calculating/verifying CRC for each outgoing/incoming data frame and also handles any hardware acknowledgements for example. The layer also handles requests for unique logical addresses as well and generates and assigns unique addresses. The MAC layer utilizes driver(s) handling the underlying physical communication channels or bus(es). As the data frames arrive on the buses, the MAC layer copies the received data into a memory pool and passes the pointer to the copied data to Link layer. At least one embodiment supports communication over multiple underlying data transfer technologies or hardware implementations such as serial, CAN, SPI, SDIO, USB, or any other type of communications medium or data bus. This structure along with an exemplary message types in FIGS. **12**A-B.

(34) In one or more embodiments, the data link layer is responsible for data integrity, for example through the use of CRC checking or any other type data integrity coding or format desired. Embodiments of the data link layer are also responsible for logical address assignment. For example, this layer is responsible for assigning and managing logical addresses of modules in a device. All the modules like Pump Motor Controller, Power Supply Controller, Communication Engine, User Interface Controller, etc., have a unique ID so that they can be uniquely identified in a pump. The protocol stack can support 254 modules as the address field is 1 Byte field and logical addresses 00, 01, and FF are reserved addresses. If modules are identified according to their unique hardware address (MAC addresses), and as the hardware addresses are more than a Byte in size, this would add overhead to the protocol. To avoid this, each module may be assigned a logical address between 1 to 255 and this layer then maintains the assigned addresses. The application layer does not need to know what the hardware address is or what the logical address is in general, which simplifies logical and API calls.

(35) One of the modules is generally assigned with the task of generating unique logical addresses for other modules in the device, no matter if those modules are connected directly to this special module or not. When the device powers on, all the modules power on as programmed. The module responsible for generating address for devices is called the "root" device. The root device is aware of its special role and assigns itself a logical address of 01. As other modules wake up, they assume their logical address as 00. They know that 00 is not a valid address but also know that there exists a module with address 01 who can provide a unique address to them.

(36) Hence, these modules send address requests to a destination with address 01. On receipt of this message, the root module checks its internal

table to verify if the requesting hardware already has a logical address assigned. If a logical address is assigned, the root module sends that same logical address in response; else it generates a unique logical address, updates this address in its internal table against the requester's MAC address and sends this address in response. On receipt of an Address Response, the requester module starts communicating with this logical address.
(37) A module in one or more embodiments may not communicate without a valid logical address. If multiple modules try to request for a logical address, there will be collisions. Due to collisions, no requests ever reach the root module, and thus none of the modules receives a logical address. In this scenario, other modules will retry after a random period of time. Depending on the criticality of device, the amount of random time can be varied, i.e. critical devices may wait for lesser period of time before a retry. The amount of wait time may be part of configuration and the devices may wait with reference to their internal clock for example.
(38) If a device does not desire to use the dynamic addressing mechanism, each module may be programmed with a unique address, for example to implement a static versus dynamic address assignment scheme. Embodiments may still utilize a root module that maintains the addresses of the connected modules.
(39) Embodiments of the data link layer may also implement routing. As mentioned, a module may have multiple bus types or topologies and there may be different type of devices connected on various buses. If a Data Link layer receives a packet that is not addressed to it, it first checks if it has multiple bus architectures and if true, it forwards the message to other buses; else it simply discards the packet. This kind of addressing mechanism is well suited for star topology for example. Hence if PMC1 wants to send data to PMC2 but there is no direct data path, then it will re-route it through the root module. In this case, the root module can broadcast the message in the network or perform a lookup in its internal table and just forward the packet on a specific line. Hence, in one or more embodiments that implement routing, each module that supports multiple communication buses may maintain a list of all devices directly connected to the module so that they can efficiently route the packets. As stack supports data routing, it seamlessly bridges multiple heterogeneous data buses, thus making communication, bus topology independent. Few examples of possible bus topologies include Ring, Star, Mesh, and Tree topologies or any other topology that may be utilized to transfer data.
(40) FIG. **3** illustrates an embodiment of an address request method implemented within the manager layer. As shown, when a device is added to the system, for example hot-swapped in, the device boots and requests an address from the root device. The new device waits for a response and if a timeout occurs, requests an address again. Once the root device receives the address request message, it looks up an available device number and generates a logical address for the new device and updates the table. Alternatively, if there are no available numbers left a NAK with appropriate error message may be returned to the new device. The root device returns the new device logical address to the new device in an address response message. Any further requests for the address are handled by lookup via the root device. The new device stores the logical address in a local table for further use. This capability generally does not exist in medical devices or infusion pumps since the configurations are generally assumed to be fixed, using a fixed operating system and fixed bus without regard to potential new devices and new types of devices that may communicate with a root device.
(41) FIG. **4** illustrates an embodiment of a simple infusion sequence utilizing various messages provided by embodiments of the method. Once the address of a new device is obtained, it may communicate with the other components within the system. The figure shows user interface controller UIC having device number 1, initially connecting to a drug infusion pump having device number 3, wherein the logical addresses of the devices, or device numbers are obtained as shown in FIG. **3**. The UIC accepts input from a Care Giver that indicates an infusion is to take place. The UIC application calculates the necessary steps to achieve the infusion and sends an infusion header and data message to the drug infusion pump, which acknowledges the message. The UIC then sends an infusion safety data message, which is acknowledged and after the infusion is complete, the UIC sends an infusion stop data message, which is acknowledged. This scenario is a typical scenario that enables any type of drug infusion pump to be added to a system and utilized, for example in a hot swap scenario where an infusion pump may return an error or a different type of drug infusion pump is to be added to the system and utilized for example.
(42) FIG. **5** illustrates an embodiment of a connection method implemented within the session layer. In the scenario shown, the UIC requests a connection in order to communicate with the PMC to command the PMC and/or for example obtain status updates. In this case, PMC acts as a service provider as the PMC is providing status updates on a known port. UIC sends a connection request to PMC on that port, e.g., port 10, shown as a message passing from left to right. After receipt of the connection request, the PMC accepts the request, generates a unique CID, e.g., 26 for this communication and updates its internal table. The PMC sends the generated CID back to UIC as a part of connection accept message, shown traveling from right to left. On receipt of connection accept message from the PMC, the UIC extracts the CID from the message and updates its internal CID table as shown in the lower left. The UIC then sends an acknowledgement message to the PMC to confirm the successful receipt of CID. If the PMC is not able to process the request from UIC and hence cannot establish communication, the PMC sends a connection reject message to the UIC. On receipt of connection reject message, the UIC may retry to obtain a connection. See also FIGS. **10**A-D, **11**A-B and **12**A-B for an embodiment of the exemplary message structures that may be utilized to form an implementation of various layers, which are described further in detail below.
(43) FIG. **6** illustrates an embodiment of a data exchange method implemented within the session layer. Once the PMC receives acknowledgement from the UIC, the connection process is complete. At this time, both devices may exchange data using the agreed CID. When the session layer of PMC receives any data from the UIC with a valid CID, it performs a lookup in its internal table against the ‘Destination ID’ and ‘CID’ to resolve the port number where the packet is to be forwarded.
(44) FIG. **7** illustrates an embodiment of a disconnection request method implemented within the session layer. On completion of data transmission, either of the communicating parties may request for a connection termination. As shown, the UIC initiates the process of connection termination. It sends a disconnect request to PMC with the respective CID. The PMC processes the request and if there is no active communication, the PMC will send an acknowledgement to the UIC and delete the CID entry from its table. On receipt of disconnection acknowledgement from PMC, the UIC also removes the CID entries from its table.
(45) Although the general session layer communication protocol has been described above, a more in-depth description of the Session layer messages follows, according to one or more embodiments of the invention. The message structures utilized in one or more embodiments of the invention as described below are shown in FIGS. **10**B-D.
(46) Connection Request Message
(47) For a connection-oriented communication session, when an application opens a socket to communicate over a port on some other device, a handshake is performed before the communication starts. The handshake begins with a connection request type message to the service provider. The “layer flag” is set for this message type. Therefore, the request packet is consumed by the session layer. The connection type may be initially set to “Unknown” suggesting that the data packet is neither connection-oriented nor connectionless. The message type is set to “Connection” as the command is used to establish new connection. The message is a request for establishing new connection; hence “Command” field has “Connection Request” set. The application requesting a connection specifies the destination's port address and also provides its own port address, hence the connection request packet has source and destination port address.
(48) Connect Accept Message
(49) On receipt of a connection request message, if the service provider has enough resource, it responds with a connection accept type of message. The service provider generates a CID for the communication and sends it to the requester as a part of this message. As the connection requesting entity has no information of the generated CID, the service provider sends source and destination port address as a part of this message to let the other end know about the generated CID.
(50) Connection Acknowledgement Message

(51) On receipt of a connection accept message, the requesting end updates its internal table with the received CID. In response to connection accept message, the requesting end sends an acknowledgement message to indicate the service provider about the receipt of CID and complete the handshake. It is possible that multiple applications on one module request to communicate with one application on another module on the same port number, e.g., many-to-one. To inform the service provider about the particular application that is sending an acknowledgement, “source port” is added to the acknowledgement message.
(52) Connection Disconnect Message
(53) Once the communication is completed, any one of the participating entities may request a connection disconnect for a graceful termination of the connection.
(54) Connection Disconnect Acknowledgement Message
(55) This message is sent as an acknowledgement on receipt of a disconnect message. The message is intended to ensure that a communication is not terminated if an active connection still exists. If a disconnection acknowledgement is not received within a certain time period, a disconnection attempt may be made again.
(56) Connection Reject Message
(57) If the service provider cannot accept any new connections, it sends a connection reject in response to a connection request message. In the connection reject message, it sends the reason for rejecting the request. On receipt of a connection reject message, the requester may retry after some time for example.
(58) CID Info Request Message
(59) Any participant involved in communication can request for status of CID. This message acts as a ping message to verify if the destination port is open and CID is an active CID.
(60) CID Info Response Message
(61) On receipt of a CID Info request, a CID Info Response is transmitted. This message contains the source and destination port addresses involved in communication, window size for transmission, etc., and also indicates if the CID is active or not.
(62) Socket Status Request Message
(63) This message is utilized to request socket related information such as the type of socket, purpose of opening this socket, etc.
(64) Socket Status Response Message
(65) This message is sent in response to Socket Status Request message. The message contains socket related information such as the type of socket, purpose of opening this socket etc.
(66) Subscribe to Service Message
(67) The communication protocol enables applications to provide a service, e.g. a broadcast service. For example, the PMC may have a service running that broadcasts PMC status periodically on a known port. If the UIC requests the PMC status, it may simply subscribe to this service with the PMC and receive the messages. Typically these services are one-way communication.
(68) Subscribe to Service Acknowledgement Message
(69) Once the service provider receives a subscription request, it has to provide a CID to the requester. The CID is delivered through an acknowledgement message.
(70) Unsubscribe from Service Message
(71) If a subscribed application no longer desires to be subscribed to a service, it may request to unsubscribe. On receipt of an unsubscribe service message, the service provider removes the entries from its internal CID table and sends an acknowledgement to the requester. If the service provider finds that there is no one subscribed to a service, it may decide to stop the broadcast service until it has at least one subscribed application.
(72) Unsubscribe from Service Acknowledgement Message
(73) On receipt of this message the application requesting to unsubscribe, removes entries of CID from its internal table and releases the involved sockets and ports.
(74) Device Address Request Message
(75) An application may request a logical address for a device using this message.
(76) Device Address Response Message
(77) On receipt of an “Address Request” message, a device sends its address as a part of the response message. Alternatively, a Device Address Response Message may be sent independently at anytime and may not necessarily be tied to a request message.
(78) Device Type Request Message
(79) This message is used to request name of a device. Every connected device has a unique address but may have non-unique names or no names. Device types can be PMC, CE, UIC, etc.
(80) Device Type Response
(81) This message is generally sent in response to “Device Type Request” message and contains the type of the device sending this message. Alternatively, a Device Type Response Message may be sent independently at anytime and may not necessarily be tied to a request message.
(82) Connection-Oriented Data Message
(83) At least one embodiment of the session layer adds just two bytes of header information when sending data between devices. The CID is generated and exchanged during the handshake process prior to data transfer.
(84) Connectionless Data Message
(85) Connectionless data transfer is used when no handshake is required to transfer data. As there is no handshake, there is no CID generated for the communication and hence both source and destination port numbers are utilized to ensure the delivery of data.
(86) FIGS. **11**A-B and **12**A-B illustrate corresponding message structures for exemplary embodiments of the Transport layer and Data Link layer respectively and are described further below.
(87) FIG. **8** illustrates a layer flow diagram that shows the flow of data within the various layers implemented in at least one embodiment of the invention. Specifically, data flow up the protocol stack for incoming data is shown. The destination application buffer location is not known until the data frame moves up to manager layer. Hence, the fragment is stored in a memory pool until it reaches manager layer and once the target application is resolved, the data is copied from the memory pool into application buffer. In one or more embodiments, memory utilization may be minimized by returning a buffer to memory if the buffer is over a predefined age threshold.
(88) Data Link Layer
(89) Data Link layer controls one or more physical communications links, data buses. The layer filters the messages directed to the specific device and ignores other messages. The layer may compute a CRC on the received packet and verify it with the received CRC. Valid data frames are copied into a memory pool and pointer to these messages are forwarded to the transport layer.
(90) The transport lower layer and data link layer run as an independent service and stores data in the designated priority queue, P1, P2, P3 or P4. The transport upper layer, session and manager layers execute in the application space, and the transport upper layer maintains pointers to the priority queues and Communication ID tables. In one or more embodiments, the memory pool, priority queues and CID tables are in shared memory space.
(91) In one or more embodiments, the data link layer is further divided into two sub-layers, a link layer and a MAC layer. The MAC layer may interface with bus drivers and has a buffer for each underlying bus. As the data arrives on these buses, the data is copied into these buffers and then forwarded to link layer. The buffer may be implemented as a pair of buffers, while one buffer is used for receiving new data, other buffer is used to

transfer previously received data.
(92) The link layer copies the data from buffers into the memory pool. The memory pool is a contiguous memory block and each memory block may be implemented as a factor of frame length. As the application consumes data, the data is removed from the memory pool to make room for new data packets. As the application consumes data randomly, there may be memory holes in the memory pool. Hence, the link layer generally maintains a list of available memory locations in the memory pool. When memory is freed from the memory pool, the pointer to available location is added at the end of this list. When a new packet arrives, it is placed at the memory pointed by the first element in this list. If there is no element in the list, memory pool will be considered full and the packets will be dropped. In one or more embodiments of the invention a memory manager may be utilized to control access to memory from the various layers, including concurrent access control of memory from the various layers. Embodiments of the invention may minimize or altogether avoid multiple copying operations by maintaining one copy of data in the memory pool while passing pointers to the memory as the data moves up and down the stack. By controlling access to the memory during access, semaphores may be utilized to ensure data integrity while allowing multiple processes to effectively utilize the data in a concurrent manner. Avoiding multiple copy operations enables minimal memory utilization in embedded environments and minimizes processor utilization as well.
(93) As the transmitter has tendencies to push data on buses, they can soon over-utilize the bus by transmitting too much data. The bus driver at the MAC layer in one or more embodiments may be implemented to handle such scenarios.
(94) Transport Layer
(95) In one or more embodiments, the transport layer may be divided into two sub-layers, a transport upper and a transport lower layer. The transport upper layer resides in application space whereas the transport lower layer resides in kernel space. These two layers together handle transport layer functionalities.
(96) The transport layer is implemented in one or more embodiments to reassemble fragmented data and also to resolve data priority. When a new data packet is received by transport lower layer, a timer may be started for the data. If the data is not consumed before the timer expires, the data may be discarded and the memory freed from the memory pool. This avoids memory starvation if no application exists to consume received data. If the acknowledgement field was set, the transport layer sends a NAK, “timed out in priority queue” error code, for example.
(97) The transport layer header has an acknowledgement flag and if the flag is set, the receiving transport layer will have to send some kind of acknowledgement for the received data fragment. If fragmented data is received, the acknowledgement is sent after receiving window size amount of data or a complete message. This flag is set for a connection-oriented data transfer to ensure delivery of data. This flag may also be set in a connectionless data transfer only if data fragmentation is utilized.
(98) Fragmented Data Packet Handling
(99) In case of fragmented data, before the transmitter starts sending any data fragments, the transport upper layer at the transmitter first requests a window size from the receiver. The window size may be exchanged once during first data transfer or may be obtained before every data transfer. Window size is the number of data fragments that can be sent before an acknowledgement can be expected. When receiver receives a window size request, transport upper layer at receivers end, computes the amount of free memory in application buffer and sends the response as window size in the ‘window size response’ message.
(100) In one or more embodiments, the transport upper layer at the transmitter side initializes a data structure for the CID that requested a window size. In this structure, the transport layer stores the CID, last reported window size, last successfully received fragment number and the maximum allowed time period between two fragments, etc. Also, the transport upper layer at the receiver maintains same structure. The transport layer expects that the fragments will be sequentially numbered starting from 1 in one or more embodiments.
(101) As the transmitter receives a window message, it calculates the number of fragments to be transmitted before expecting an acknowledgement. The transmitter starts sending data fragments in sequence starting from fragment number 1 for example.
(102) When the receiver receives first fragment, the transport lower layer starts a timer on the received data frame and places the fragment it into the respective priority queue. The transport upper layer updates the structure and stores the sequence number of the fragment. If the fragment is delivered to the application buffer by upper layers, the upper layers inform the transport upper layer about the success. The transport upper layer updates its structure with the first fragment being delivered. Upper layers do not inform application about the available fragment until all the fragments constituting to a message are received. An application buffer is used for re-assembly of fragments to minimize memory footprint.
(103) If the transport upper layer receives all the fragments for a window successfully, it waits for all the fragments to be delivered to application buffer successfully. Once all the fragments are sent to application buffer, the received fragment number and delivered fragment number match and the transport upper layer sends an acknowledgement for the last fragment in the sequence. The transmitter receives the acknowledgement at the transport upper layer.
(104) Ideally, the transport layer accumulates all fragments, verifies that they are in sequence and merges them into one complete message before sending it up the stack. However, in one or more embodiments, the transport upper layer forwards the frames to the session layer as they are received, but ensures that the fragments are delivered in sequence. This optional implementation may be utilized to lower memory utilization. This is the case since the message does not have to be reconstructed in full within the stack until the full message is received in the application. As the fragment number in the transport header is 10 bits wide in one or more embodiments, the layer can support a maximum of 1023 fragments (fragment number 0 is reserved and represents a non fragment data frame) before the fragment numbering overflow. As each fragment has a maximum of 248 Bytes payload, hence a total of 253,704 Bytes is required at the receiver end for each active connection to accommodate all the fragments. Any other size of fragment number field may be utilized to increase the overall size as one skilled in the art will recognize.
(105) At the receiver, as the fragments are received, transport upper layer updates the last fragment number in its structure. Before updating, it verifies that the received fragment is in sequence with previously received fragment. If it detects a missing fragment, the layer still forwards the fragments up the stack, but in their respective token puts an offset value. Metadata along with a pointer to the received data fragment is called a token. This offset value is used by manager layer to provide a gap while accommodating other fragments around the missing one, so that the gap can be filled once the missing fragment is received. For example to create an empty space in memory so that when the missing frame is finally received, it will be accommodated in this empty space to complete the final message. Meanwhile, transport upper layer waits for the fragments to arrive and then looks for any missing fragment in the sequence. Once the layer generates a list of all missing fragments, it requests for retransmission of fragments from the transmitter. Once the missing fragments are received, they are forwarded to upper layers so that they can be used for filling the empty spaces in final message.
(106) When retransmission is required, transport upper layer at receiver end, sends retransmission request message with the desired fragment number in it. The receiving end maintains a list of missing fragments and as the missing fragments are received, their entry is removed from this list.
(107) If the transmitter retransmits an already transmitted fragment, the receiver compares the fragment number with last received fragment number and will detect that there has been a retransmission. The layer checks if the retransmission was requested by the receiver explicitly or not. If the retransmission was intentional, the fragment is consumed else the fragment is dropped assuming a false retransmission of data.
(108) Once the transmitter sends one window size worth of fragments, it starts a timer and waits for an acknowledgement on the last fragment in the sequence. The transmitter may send any further fragments only when it receives an acknowledgement. If the acknowledgement is delayed and the timer expires, the transmitter may send a “window size” request message before retransmitting the fragments. A receiver may fail to send an acknowledgement if the receiver is too busy or its buffers are full. Hence, a “window size” message is sent because it serves two purposes, the first being that a response to this message implies that the receiver is ready for accepting messages, and the second being that the new responded window size buffer is available at receiver so that chances of getting an acknowledgement increases.

(109) In case of missing fragments, the receiver sends a retransmission request instead of an acknowledgement. A retransmission request can only be sent if the last fragment in the sequence was either received successfully or was found missing. Hence, the transmitter considers a retransmission request message as an implied acknowledgement and no more waits for an explicit acknowledgement, but may wait on acknowledgement for retransmitted fragment.
(110) Missing fragments can be of three types, the first fragment missing, any fragment(s) missing between first and the last fragment of a complete message, and the last fragment itself missing. If the first fragment is missing and the receiver starts receiving from fragment number 2, it accumulates all the messages till it receives window size messages and explicitly requests for the 1st fragment. The same technique is used for requesting any missing fragment between 1st and last fragment.
(111) Missing the last fragment of a complete message may be a complicated scenario because transmitter never informs the receiver about total number of fragments needed to send a message and hence, there is no way for receiver to know when the message completes. Missing “last” fragments can be of two types, missing the last fragment from a window and missing the last fragment of a message. In the case of missing the last fragment from a window, it is easy to detect. Every time a fragment is received, the receiver starts a timer and waits for next fragment to be receive before the timer expires. The transmitter sends the last message for the window and waits for an acknowledgement. If this message is lost, the receiver waits for this last fragment to arrive. The timer at the receiver expires earlier than the timer at the transmitter. As the receiver keeps track of fragment sequences and window size, it realizes that the last fragment was not received on time and hence sends a retransmission request for the last fragment.
(112) A more difficult problem occurs when the last fragment of a message is lost. As the receiver has no idea about how many fragments will constitute a message, it looks for the fragment with ‘last fragment’ flag set. This fragment indicates the receiver that it was the last fragment from the message. If this fragment is lost, the receiver has no idea when to stop reassembling fragments. To ensure delivery of this last fragment, the transmitter can use following two approaches.
(113) In the first approach, the transmitter knows that the last fragment is approaching. It explicitly reduces the window size to make sure that the last fragment of the message becomes the last fragment of the window as well. As the receiver can detect the last fragment from a window, if the last fragment from a message is lost, the receiver may request retransmission.
(114) In the second approach, the transmitter will send the last fragment with ‘last fragment’ flag set, followed by few fragments with random payload but with incremental fragment number. If the last fragment of the message is missing, the receiver will detect the missing fragment as there will be gap in sequence numbers and will request for retransmission. When the receiver attempts to arrange the fragments in sequence, it detects the fragment with ‘last fragment’ flag set and hence discards all fragments following this fragment.
(115) Non-Fragmented Data Packet Handling
(116) For a non-fragmented data frame, it is first received by transport lower layer, which starts a lifetime timer on this frame and puts the frame in appropriate priority queue. The frame is picked from the priority queue by transport upper layer, which forwards it to other layers, for example Session layer.
(117) Session Layer
(118) Session layer major responsibilities are to ensure application-to-application delivery of data and generate unique CID's within a system. The stack works on the principle of service provider and service consumer. The application providing service generates unique CID's for the engaged participants. The CID is unique within the system running the service provider application in one or more embodiments. The CID may be thought of as a key used to hide the information about source and destination ports engaged in communication.
(119) The session layer may be implemented in a lightweight or a very thin layer to a connectionless communication because a connectionless data packet will contain the source and destination port addresses as part of their headers and hence does not utilize a CID.
(120) Packets reaching the session layer may be divided into two categories, namely data and control. Further, the incoming data can be connection-oriented or connectionless and fragmented or non-fragmented.
(121) Connection-Oriented Data Transfer
(122) Connection-oriented data transfer makes use of a connection through a handshake process. After an initial handshake process is complete as is described further below, data exchange occurs. In connection-oriented data transfer, embodiments of the invention utilize a data header with an acknowledgement flag set and connection type set to 01, for example. Data being exchanged may be fragmented or non-fragmented based on the size of the data and the underlying packet size supported by the physical medium.
(123) Fragmented Data
(124) When an application writes to a virtual port, the session layer adds a session layer header to the data and forwards it down the stack. In one or more embodiments, the session layer header is 2 bytes wide. Hence, if fragmentation is needed at the transport layer, the first fragment is set to contain the CID from the session layer while the rest of the fragments may contain only application data. The session layer at the receiving end forwards the first fragment that contains the session layer header, but is unsure as to where to forward other fragments from the sequence as there is no CID information in subsequent headers. Also, if two or more applications on one device want to send data to one device, it is not possible without further information in general at the receiving end to aggregate fragmented data because there is no way to uniquely identify which application is sending what data fragment. To resolve this issue, the transport layer copies session layer header to all the related fragments. As all the fragments will now contain CID, they can be uniquely identified at the receiving end.
(125) The session layer header contains an acknowledgement flag that is utilized in the case of complete messages. As the session layer ensures application-to-application delivery of data, it sets the acknowledgement flag for the receiver to acknowledge successful delivery of data. As the header is copied in each fragment, the session layer will look at the flag and will acknowledge the transmitter every time a fragment is delivered which is not what acknowledgements are generally for, i.e., a complete message acknowledgement.
(126) To avoid this issue, the transport upper layer at the receiver end appends metadata to packets as they are sent up the stack. Metadata along with pointer to received data fragment is called a token and instead of passing data, transport layer passes a token to session layer. In the case of exceptions in behavior of the session layer, metadata provides guidelines for the session layer to follow. For example, the session layer will not send any acknowledgements for data fragments, and when the transport upper layer receives a fragment with a “last fragment” flag set, it updates the metadata so that session layer knows that it needs to send an acknowledgement to the transmitter regarding the receipt of a complete message.
(127) Flow of Control
(128) As the fragments move through the session layer, session layer extracts the CID from the fragments, performs a lookup in the Communication ID table based on CID and the sources logical address obtained from the metadata. The session layer determines the associated file descriptor source and destination ports for the CID. Once the file descriptor is known, it removes all the headers and modifies the metadata to communicate the file descriptor detail to manager layer.
(129) Once the fragment arrives at the manager layer, the manager layer extracts the file descriptor information from metadata and forwards the fragment to respective application. Before the manager layer forwards the message to the application, it determines if the file descriptor is still in use and in the state of accepting data. If conditions are favorable, the message is copied into the application buffer and a “message received” flag in file descriptor is set. If the current operation on the file descriptor is a blocking read, the read function call returns with number of bytes available in application buffer. If the current operation is a non blocking call, the application either checks the flag and if set, reads data from buffer, or the manager layer may make an asynchronous function call on receiving data.
(130) After delivering the data to the application, the manager layer returns the token to session layer. This token contains information about the state

of the previously passed message. Depending on the state of token, the session layer performs activities such as sending a session-to-session layer acknowledgement.
(131) If the data is fragmented, session layer further modifies this token and sends it down to transport layer, otherwise the session layer consumes the token. The transport layer determines if the fragments were delivered in sequence they were sent and accordingly controls acknowledgements and window sizes.
(132) Non-Fragmented Data
(133) If a message size is less than the Maximum Transmission Unit (MTU), no fragmentation is required and the complete message is sent in one frame. As the frame moves up the stack, transport upper layer adds very little information to the metadata as complete information for the session layer is already available in the frames header. The session layer reads the header and extracts the data type. If the data type is connection-oriented data, the session layer extracts the CID and performs a lookup in the CID table to determine source and destination ports. The session layer removes all the headers from the datagram, updates the metadata with the destination file descriptor, and forwards it to the manager layer.
(134) Connectionless Data Transfer
(135) As mentioned above, in a connectionless data transfer, the session layer may be implemented in a lightweight or very thin layer. As connectionless data transfer does not utilize a handshake, no CID is generated. Due to the absence of the CID, the protocol header utilizes source and destination port addresses. The session layer reads the destination port address and determines the associated file descriptor and forwards the message to that port. As connectionless data transmission does not guarantee delivery of data, the acknowledgement flag on the frames is set to false.
(136) If a connectionless data frame is larger than the MTU, the transport upper layer fragments the data into manageable sizes without setting the transport layer acknowledgement flag as would be done in connection based communications. During reassembly, if transport layer sees any missing fragments, it discards the complete message. Through a token, the transport layer informs upper layers to discard previously accumulated fragments in application buffer.
(137) Manager Layer
(138) Manager layer handles file descriptors and forwards packets from lower layers to appropriate file handlers. The manager layer also performs the copying of data from the memory pool into the application buffer. The manager layer knows the size of the application buffer and the application buffer size may be smaller than one frame length.
(139) If the application buffer is large enough, the manager layer copies the complete message into application buffer. If the application buffer is not large enough, the manager layer copies data in a sequential manner. The manager layer fills the application buffer with data and waits for the application to read the data before copying the next portion of data. Once data is successfully delivered to the application, depending on the token, the manager layer informs the session layer regarding success.
(140) Control Flow Up the Stack
(141) The flow of control is now described as data moves up the stack from the lowest layer to the application layer.
(142) Data Link Layer
(143) The data link layer control is described with respect to the two sub-layers that make up the data link layer, namely the link layer and the MAC layer. The MAC layer controls the physical bus drivers.
(144) MAC Layer
(145) As the datagram arrives on the physical bus, the bus driver copies the datagram into a buffer. Once the complete datagram is available in the buffer, the MAC layer calls an API in Link Layer to copy the available data into the memory pool.
(146) The link layer API returns a value to indicate the outcome of the copy operation. The operation may succeed or fail. The returned error code provides the reason for any failure. The MAC layer waits for the API to finish the operation before storing newly available data into the buffer.
(147) Link Layer
(148) As discussed in the sections above, the memory pool may be fragmented due to applications consuming data at random rates, resulting in holes in the memory pool. In one or more embodiments, the link layer maintains a link list, or a doubly link list, or bit map or any other data structure capable of storing available memory locations in the memory pool. When a memory location is made available, a pointer to the memory location is added to the tail of the list. When a new datagram is available, it gets copied at the memory pointed by pointer in the head of the list. Though the received message can be of any size and wherein a maximum size exists, for example 256 bytes, the size of the memory pool is selected to be an integral multiple of the maximum datagram size. This simplifies memory management, as the stack is aware of the size of allocated memory given the pointer to that memory. There may be instances when a datagram is available at the time when memory is made available in the memory pool. In this case, both the copy and the delete processes will try to access the list simultaneously leading to concurrency issues. In one or more embodiments, the memory pool may include non-uniform size buffers for a more flexible buffer implementation at the cost of memory management complexity as one skilled in the art will recognize.
(149) When the MAC layer calls an API to copy the data from hardware buffer to memory pool, the API first checks the list for any available memory location in the pool. If memory is available, the API copies the datagram to the memory location pointed by the head of the list and deletes the pointer from the list. If no space is available, for example the link list is empty, or error occurs during the copying to memory pool, the API returns respective error code.
(150) After successfully copying the datagram, the API adds the pointer to the datagram in a list with a number of timer ticks remaining before the data should be delivered to application. This API may be reentrant as the MAC layer may be riding over multiple bus architectures and the data may be available in multiple buffers at the same time resulting in calling this API while the layer is still servicing the previous call.
(151) The protocol stack may be implemented with a time limit within which a datagram is to be used by an application, or else the datagram is dropped from the memory pool. To enable this feature, embodiments may implement a global list containing pointers to each datagram with the timer count on each pointer. As the new packets arrive, an API adds the pointer to this packet at the end of this list. The API adds “time to live” value to the current timer count and generates a timer count that represents an expiration time for the packets. When timer count changes, an API looks at the timer count starting from top most element in the list and starts deleting datagram if their timer counts are less than or equal to current timer count.
(152) Once the data is consumed by the application or the data times out, an API is called to remove the datagram from the memory pool and add the pointer to the available memory list. This API may be reentrant as the data may expire at the same time it was consumed by the application. Both processes may attempt to delete the same datagram, therefore semaphores/locks may be utilized to effectively serialize control.
(153) When data gets copied to memory pool, the link layer generates a token for the packet. The token contains the pointer to the datagram and length of the datagram. This token is forwarded to the transport layer through a transport layer API for further processing.
(154) Transport Layer
(155) After the transport lower layer receives a token, the transport lower layer determines if the frame is a transport-layer-to-transport layer message. If the ‘layer flag’ is set, then these types of messages are layer-to-layer messages and hence are not forwarded to upper layers. If the flag is not set, transport lower layer looks at the priority of the message and places the token into appropriate priority queue.
(156) In one or more embodiments, the transport upper layer receives the token from the priority queue and determines if the ‘extended flag’ is set or not. If the flag is set, it indicates that a large volume of data is to be expected and informs the API that an extra byte has been used in header for sequencing large number of fragments.
(157) The layer also reads the “Last Fragment” flag. A set ‘last fragment’ flag indicates to the layer that the current datagram fragment is the last fragment in the sequence of fragments and hence the end of one message. If there is any fragmentation, at least one fragment will have this flag set.

(158) The layer further reads the acknowledgement flag. If the transmitter requests or otherwise is to be sent an acknowledgement for delivery of the datagram to the receiver's transport layer, the layer will set this flag and the receiver will acknowledge the receipt of the packet. If the devices engaged in communication have agreed on a window size for acknowledgements, then the transport layer acknowledges after receiving window size messages else the layer acknowledges each datagram.
(159) The transport upper layer adds more information to the data token and forwards it to session layer. The transport upper layer informs the session layer if the message is a complete message or not. In case of fragmented message, the transport layer informs the session layer about receiving the last fragment, so that session layer may send an acknowledgement if needed.
(160) Session Layer
(161) From the data pointer in the token, the session layer accesses the frame and extracts session layer header. From the header, session layer first determines if the message is a layer-to-layer message or needs to be forwarded up the stack. If the message is a layer-to-layer type message, then the message is consumed by session layer.
(162) If the layer flag is not set, the frame is forwarded up the stack. The session layer reads the ‘Connection Type’ field and determines if the message is of unknown connection type or connection-oriented or connectionless. An unknown connection type is generally for the messages exchanged during handshake process, whereas a connectionless message does not need an acknowledgement for delivery, and connection-oriented messages are the ones that use an acknowledgement on successfully delivery.
(163) The session layer further looks into the message type field to determine the type of frame. The frame type is used to determine the purpose of the frame, and only ‘Data’ type frames are forwarded up the stack and the control type frames are consumed at session layer.
(164) The CID is generated by the application providing a service. Any application that wants to use the service will request a communication ID. CID is unique within one module, for example all of the CID's generated by the UIC are unique within a particular UIC. The CID is generated through a handshake process, where the application using the service sends the details required for uniquely identifying an active connection and receives the CID in response.
(165) The CID specifics and details may be stored in a CID table located in a shared memory region in one or more embodiments, so that the session layers of all the applications may access the CID. In a connectionless data frame, there is no CID information as there is no handshake utilized to establish a connection. Hence connectionless frames contain both source and destination port address in the header.
(166) In a connection-oriented data transfer, there exists a CID in the session layer header. Once the session layer determines the CID from the header, the layer combines the information with the source logical address available in the data token to uniquely identify an entry in CID table. From this table, the session layer determines the source and destination port address and the file descriptors handling the port. The source logical address of the received frame is set by the data link layer along with the file handler information and is forwarded in the data token to the manager layer.
(167) If the received frame is connection-oriented and is a complete message, the session layer maintains a record of the message and forwards the data token to the manager layer. Once the manager layer copies the frame from memory pool into the application buffer, the manager layer notifies the session layer about the successful delivery of data. On receipt of notification, the session layer sends an acknowledgement to the transmitter session layer regarding the successful delivery of data. If the delivery was unsuccessful, as a part of the acknowledgement, the session layer forwards the error message returned from the manager layer to the transmitter.
(168) Manager Layer
(169) The session layer calls an API in manager layer and passes the data token to the manager layer. The manager layer copies the data from memory pool into the application buffer and notifies the session layer regarding the copy. The manager layer notifies the lower layer about the delivery of message by modifying the data token and sending the data token back to the session layer. Once the data is successfully copied, the manager layer removes the frame pointer from the list of frames monitored by the timer and deletes the frame from the memory pool to make room for new packets.
(170) It may happen that the application buffer is smaller in size than the received data frame, in such cases the manager layer will fill the application buffer with what it can hold and wait for the application to consume it. Once the application consumes the message, the remaining portion of the message is copied and the process is repeated until the complete frame is consumed. Before starting the progress of copying messages in small sizes, the manager layer removes the pointer to the frame from the timer-monitored list because the timer may expire and corrupt the message. Also, the manager layer notifies the lower layer regarding successful delivery of data only when a complete message is sent to the application. At the end of the sequential copy process, the manager layer deletes the frame from the memory pool.
(171) In the case of fragmented data, as the fragments are received by this layer, it copies the fragments into the application buffer and notifies the session layer. The session layer forwards the notification to the transport layer. The transport layer, after receiving notifications for a window size number of messages, sends an acknowledgement to the transmitter about receiving the messages. When the last fragment is successfully delivered to the application, it implies that one complete message was delivered. In such cases, the manager layer notifies the session layer of success, and the session layer sends an acknowledgement message to the transmitter regarding the success, thus providing guaranteed delivery of data.
(172) Data Flow Down the Stack
(173) Assuming that in case of a connection-oriented data transfer, the handshake process has already been done and a valid CID has been already generated, the application copies data into an application buffer and passes a pointer to the API exposed by the manager layer for sending data over virtual ports. The application also specifies the file descriptor that handles the communication and the size of data to be written on the virtual port.
(174) The priority of a message is determined by the priority of the virtual port being used or priority can be set for the message passing through. Hence, through a set of API's, the manager layer informs the session layer about the priority of data, size of data, file descriptor for the communication, pointer to application buffer, and if data is connection-oriented or connectionless. If the data is connectionless, the session layer looks into the file descriptor table and determines the port number associated with the file descriptor. The session layer then adds source and destination port addresses as header to the data. If the transfer is to be connection-oriented, the session layer performs a lookup in the CID table and determines CID associated with the file descriptor and adds this CID as header to the data. The session layer then forwards this pointer to the transport layer and waits for an acknowledgement from the receiver.
(175) The transport upper layer determines the size of the data and determines if fragmentation is required or not. If fragmentation is needed, the transport upper layer breaks the data into manageable sizes and adds information to the header so that the data can be reassembled at the receiver's transport upper layer. If fragmentation is not needed, the transport upper layer still adds some information in one or more embodiments. For example, the transport upper layer copies the data from application buffer into transmitter memory pool and depending on the priority of data, stores the pointer into appropriate message queues.
(176) The transport lower layer eventually reads the pointer from the priority queue and forwards it to the link layer. The link layer determines the destination logical address and adds it to the data header, computes a CRC on the frame and adds it to the frame before sending it. The MAC layer determines the bus over which the destination is available and sends the data over that bus.
(177) Flow of Data Up the Stack
(178) As the data frame arrives at the underlying bus, the MAC layer determines if the frame is for the subsystem or for some other subsystem. If it is for some other subsystem, the MAC layer drops the data frame. The MAC layer copies valid data frames into a shared memory region and calls an API in the Link layer to inform it about arrival of new data. Throughout the stack, only the pointer to this data is updated to reduce multiple copying of fragments.

(179) The link layer computes the CRC on the received frame and compares the computed CRC with the CRC on the received frame. Frames with invalid CRC's are dropped. Pointers to valid frames are forwarded to the transport lower layer.
(180) The transport lower layer reads the priority of the frame and adds a pointer to the frame to the respective priority queue. The pointer to the frame remains in the queue and waits for appropriate application to consume it. Eventually, the target application's transport upper layer reads the pointer to the frame from the priority queue.
(181) The transport upper layer looks at the headers to determine if the data is fragmented or a complete message. If the data is fragmented, the layer reassembles all the messages from the sequence and then forwards it to the application layer. If the data is not fragmented, it directly forwards the pointer to the frame to the session layer through appropriate API calls.
(182) The session layer looks at the frame headers and determines if the message is of type connectionless or connection-oriented. If the message is connectionless, the session layer looks at the destination port number and determines the file descriptor handling that port. The session layer forwards the pointer to the manager layer with appropriate file descriptor information. If the frame is connection-oriented, the session layer reads the CID and determines the file handler handling that communication. The session layer then forwards the file descriptor information to the manager layer and waits for an acknowledgement from the manager layer. The manager layer sends an acknowledgement indicating whether the data was delivered to the application or not. This information is used by the session layer to acknowledge receipt of data.
(183) The manager layer may be implemented with a lightweight or thin layer and is responsible for copying the data from the memory pool into the application buffer and freeing the memory pool. Once the data gets copied into the application memory, the manager layer informs the application about data being available. The manager layer sends an acknowledgement to the session layer. Thus, to the applications, the manager layer offers synchronous and asynchronous methods for reading and writing to virtual ports.
(184) FIG. **9** illustrates an activity diagram showing routing between various devices. As shown, Device A is connected directly to Device B, which is directly connected to Device C. Device A is not directly connect to Device C. When Device A attempts to send a message to Device C, it sends the message out and Device B reads the message, determines that the message is not for the device and checks to see if there is a path to the device in Device B's destination table. If so, Device B forwards the message to Device C, which processes the data. Hence, embodiments of the invention enable routing and daisy chain or multi-bus configurations that are generally not as flexibly possible in medical devices such as infusion pumps.
(185) An embodiment of the manager layer API is detailed below. The manager layer provides the API to enable socket programming over the protocol stack. The manager layer API calls are utilized by any application that wishes to transfer data using an embodiment of the invention. pro_socket—creates an unbound socket in a communication domain, and returns a file descriptor that can be used in later function calls that operate on sockets. int16 pro_socket (ConnectionType type, uint8 *pSocket)
Arguments: type: specifies the type of socket to be created (CONNECTIONTYPE and CONNECTIONLESSTYPE for connection oriented and connection-less data exchange). pSocket: integer pointer to return newly created socket.
(186) TABLE-US-00002 typedef enum ConnType { CONNECTIONTYPE=1, CONNECTIONLESSTYPE=2, RAWTYPE=3 } ConnectionType;
(187) On successful completion, the function shall return a SUCCESS; else appropriate error code is returned. The API returns allocated socket in the reference variable pSocket passed as a parameter. pro_bind—assigns a local address to a socket identified by file descriptor socket. int16 pro_bind (uint8 uint8Socket, const ProSockaddr *pAddress)
(188) TABLE-US-00003 typedef struct pro_sockaddr { uint8 address; uint8 portNo; uint8 priority; uint8 flags; uint32 timeout; datafilter *filter; } ProSockaddr; address: holds logical address of device portNo: holds port number for connection priority: holds the priority of the port. All the data passing through this port inherits ports priority flags: holds configuration flags for changing behavior of socket TIMEOUT: flag is set, waits for an operation to complete within a given period of time, else returns. SO_LINGER: set flag indicates that a connection will be terminated only when all the data pending to be sent is sent successfully. FILTER_DATA: set flag indicates that the data matching supplied filter pattern will only be forwarded to callback function registered to handle it. If flag is reset, data matching the filter will be sent to both, regular socket handler as well as to the registered callback function. timeout: If the TIMEOUT flag is set, timeout value is specified here. A timeout value of 0 returns immediately. filter: link list of datafilter type structure defining the filter to be applied on received data. More than one element in this link list will have an ORing property. As an example, if an application wants to process data either from PMC, or UIC, or CE or all three, it will create three nodes in this link list one for each PMC, UIC and CE.
(189) Datafilter structure is used to allow applications to select what messages they want to receive, and which function should handle what type of messages. A regular expression pattern is used to create a filter on received data and once a match is found, data is forwarded to registered callback function.
(190) TABLE-US-00004 typedef struct { char *regEx; uint8 index; uint8 length; void *callback; datafilter *filter; }datafilter; regEx: pointer to regular expression to be used for matching. index: location to start looking for match in the data section of received message. A ‘0’ in this field indicates that the match will start from the beginning. length: staring from the provided index, indicates the length of data section to be considered for regular expression matching. If index contains ‘0’ and length contains ‘0’, match will be performed over the entire data section. callback: function to be called in case of a successful match. If this field is set to null, all the matching data packets will be dropped depending on FILTER_DATA flag. filter: linklist of any additional filter to be added over existing filter. If this filed is contains additional filters, on a successful match, callback is made to the function specified in the structure containing this linklist. This link list of filters has anding properties, i.e. a match is successful only if all the regEx specified in all the filters match. As an example, if an application wants to process data containing expressions PMC, UIC and CE, it will instantiate this filter for PMC and have a link list containing filters for UIC and CE respectively.
(191) One or more embodiments support three priority levels for messages namely high, medium and low. The enum defining message priority is as follows.
(192) TABLE-US-00005 Typedef enum ProPriority { HIGHPRIORITYTYPE = 1, MEDIUMPRIORITYTYPE = 2, LOWPRIORITYTYPE = 3 } MessagePriority;
Arguments: uint8Socket: file descriptor of socket to be bound pAddress: pointer to ProSockaddr struct containing address to be bound to the socket.
Return Value:
(193) Upon successful completion, the function shall return SUCCESS, otherwise appropriate error code. pro_connect—attempts to connect a socket to the specified address. int16 pro_connect (uint8 uint8Socket, const ProSockaddr *pAddress)
Arguments: uint8Socket: socket to be connected to specified address. pAddress: pointer to structure pro_sockaddr containing peer address.
Return Value:
(194) Upon successful completion, the function shall return SUCCESS; otherwise returns appropriate error code pro_listen—marks the socket referred to by “socket” as a passive socket, that is, as a socket that will be used to accept incoming connection requests using accept( ) int16 pro_listen (uint8 uint8Socket, uint8 uint8Backlog)
Arguments: uint8Socket: file descriptor of a socket that needs to be put in accepting connections mode. uint8Backlog: set a limit on number of outstanding connections in the socket's listen queue. A zero would set the queue length to system defined minimum queue length.
Return Value: Upon successful completion, the function shall return SUCCESS; otherwise, appropriate error code is returned. pro_accept—extracts the first connection on the queue of pending connections, creates a new connected socket with same socket type protocol and address family as the specified socket, and returns a new file descriptor for the socket. int16 pro_accept (uint8 uint8Socket, ProSockaddr *pAddress, uint8 *pClientSocket)
Arguments: uint8Socket: file descriptor associated with socket. pAddress: Either a NULL pointer, or a pointer to ProSockaddr struct where the address of connecting socket shall be returned. pClientSocket: pointer to an unsigned integer for returning file descriptor associated with the newly

created socket.
Return Value:
(195) Upon successful completion SUCCESS is returned along with an associated file descriptor for the newly created socket, on failure, returns appropriate error code. pro_send—initiates transmission of a message from the specified socket to its peer. The pro_send( ) function sends a message when the socket is connected. int16 pro_send (uint8 uint8Socket, const void *pBuffer, uint32 intLength, uint32 *pBytesSent)
Arguments: uint8Socket: socket's file descriptor pBuffer: points to buffer containing the message to send. intLength: length of message in bytes. pBytesSent: pointer to an integer for returning actual number of bytes sent.
Return Value:
(196) Upon successful completion, the API returns SUCCESS else appropriate error code is returned. pro_recv—receives a message from a connection-mode or connectionless-mode socket. It is normally used with connected sockets because and does not provide the source address of received data to the application. int16 pro_recv (uint8 uint8Socket, void *pBuffer, uint32 uintLength, uint32 *pBytesReceived)
Arguments: uint8Socket: file descriptor of socket. pBuffer: pointer to the buffer where message should be stored. uintLength: length in bytes of the message to be received. pBytesReceived: pointer to an integer for returning number of bytes actually received.
Return Value:
(197) Upon successful completion, SUCCESS is returned along with number of bytes in the reference passed as a parameter, else returns appropriate error code. pro_close—deallocates the file descriptor and makes the file descriptor available for functions which allocate file descriptors. All outstanding record locks owned by the process on the file associated with the file descriptor are removed. Causes the socket to be destroyed. If the socket is in connection-oriented, and the SO_LINGER option is set for the socket with non-zero linger time, and the socket has un-transmitted data, then pro_close( ) blocks for up to the current linger interval for all pending data to be transmitted. int16 pro_close (uint8 uint8Socket)
Arguments: uint8Socket: file descriptor of socket that needs to be closed.
Return Value:
(198) Upon successful completion, function shall return SUCCESS; otherwise appropriate error code shall be returned.
(199) One skilled in the art will recognize that in addition to the exemplary API illustrated above for the Manager Layer, API's for the Session, Transport and Data Link Layers may be implemented as desired to communicate the messages shown in FIGS. **10**A-D, **11**A-B and **12**A-B depending on the desired application.
(200) One or more embodiments of the invention may be implemented as a system or method. For example at least one embodiment may include a medical device communication method that includes accepting a request by a programmable device to obtain a device identifier associated with a transmitting device associated with the request, a connection type of connection-oriented or connectionless-oriented, and a receiving device number associated with a receiving device to transmit a message to. The method may also include determining a port number of a port to transmit said message to, for example either via a requesting programmable device or the programmable device that receives the request. Embodiments may also include generating a communication identifier or CID for at least the advantages stated throughout this disclosure. Embodiments may also include accepting a request associated with a medical function, inserting the CID and the medical function into the message, determining if the connection type is connection-oriented or connectionless and transmitting the message to a medical device. This scenario is shown with exemplary values to demonstrate the previous message formatting and transfer in FIGS. **13**A and **13**B, which are intended to couple with one another on the right side of FIG. **13**A and the left side of FIG. **13**B.
(201) Embodiments may also include transmitting the message to the receiving device even if the receiving device is not directly connected to the transmitting device. This enables built in routing that allows for devices to pass through messages without requiring a master to control all phases of communication for example.
(202) Embodiments may also include accepting a multicast request configured to enable multiple receiving devices to receive the message. Embodiments may further include accepting a priority parameter configured to enable prioritized handling of the message. This enables messages with high priority to be delivered before other lower priority messages and in one or more embodiments may be implemented with a plurality of message data structures such as queues, linked lists or any other data structure or structures. Embodiments may include transmitting messages from a high priority message queue before transmitting data from a low priority message queue. Other embodiments may apply any type of strategy pattern to the delivery process, and may for example change strategies depending on the type of messages that are likely to be received in particular time periods. This enables predictive handling and processing of messages to provide intelligent and robust delivery of medical functions.
(203) Embodiments may also include determining if a size of data to transfer is larger than a predetermined fragmentation value and packing the data in a plurality of messages to facilitate transfer. Embodiments may efficiently utilize memory and for example reduce latency by copying a pointer to the message between a plurality of message layers without copying the message itself. This is the case since the message does not have to be reconstructed in full within the stack until the full message is received in the application. Furthermore, embodiments of the invention may utilize optimized memory management that includes requesting memory from a buffer that includes non-uniform sizes to further increase efficiency of data memory utilization and lower overall required memory. When sending data packets or message that are larger than the maximum size allowed by the underlying hardware, embodiments may include setting a last fragmentation flag in a final message of fragmented message, starting a timer for an acknowledgement and retransmitting the final message if said timer expires. Further increases in efficiency may be achieved by embodiments that include receiving a request to change a window size for receipt of fragmented messages and adjusting memory usage based thereon, for example having lower window sizes for more reliable communication links. Embodiments may also include providing the device identifier to a new medical device that replaces the medical device after hot-swapping the new medical device for the original medical device, i.e., if a failure occurs. This allows embodiments of the invention to provide robust functionality and transparent replacement of hardware without interrupting medical functions or at least minimizing the interruptions. Embodiments may also include providing a pointer to a complete message after receipt of multiple fragmented messages without copying received message data after receipt thereof. This enables incoming data to be inserted into a buffer once and given to the application after the data is received without extraneous copying for example, which reduces memory utilization and programmable device processing required. One or more embodiments of the invention may include accepting an infusion request associated with an infusion related medical function. Any other type of medical function is in keeping with the spirit of the invention.
(204) Embodiments of the system may include a programmable device configured to accept a request to obtain a device identifier associated with a transmitting device associated with the request, a connection type of connection-oriented or connectionless-oriented, a receiving device number associated with a receiving device to transmit a message to. Embodiments of the system may further determine a port number of a port to transmit said message to, generate a communication identifier or CID and accept a request associated with a medical function. The system may also insert the CID and the medical function into the message, determine if the connection type is connection-oriented or connectionless and transmit the message to a medical device. Embodiments of the system may also implement all functionality of the method previously described and may utilize any of the data structures or API's described herein in combination.
(205) While the invention herein disclosed has been described by means of specific embodiments and applications thereof, numerous modifications and variations could be made thereto by those skilled in the art without departing from the scope of the invention set forth in the claims.

Claims

1. A medical device communication system for connecting a first medical device system with a second medical device system, the medical device communication system comprising a controller configured to: receive a first message comprising a request for an address from a first medical device system; determine in a stored table that there is no entry corresponding to the first medical device system; generate a connection identifier for the first medical device system based on the determination; and transmit the connection identifier to the first medical device system, wherein the first medical device system is further configured to update its identity with the connection identifier.
2. The communication system of claim 1, wherein the controller is further configured to facilitate transmission of a second message from the first medical device system to an additional hardware, wherein the first medical device system and the additional hardware are not directly connected.
3. The communication system of claim 2, wherein the second message is transferred between a plurality of message layers by copying a pointer and without copying said second message itself.
4. The communication system of claim 2, wherein a session layer communication is made independent of bus topology.
5. The communication system of claim 1, wherein the controller is further configured to transmit the connection identifier to a second medical device system that replaces the first medical device system.
6. The communication system of claim 1, wherein a fragmented message is reassembled into a complete message in an application buffer.
7. The communication system of claim 1, wherein the controller is further configured to: determine that a size of data to transfer is larger than a predetermined fragmentation value; and pack said data in a plurality of messages independent of an underlying data bus width.
8. The communication system of claim 1, wherein the controller is further configured to request memory from a buffer comprising non-uniform sizes.
9. The communication system of claim 1, wherein the first medical device system is connected with the controller without altering an application of the first medical device system.
10. The communication system of claim 1, wherein the controller manages connection requests from a plurality of medical device systems.
11. A method for connecting hardware components of an medical device system, the method comprising: transmitting a first connection identifier to a controller of a first medical device system responsive to a first message corresponding to a first request for an address from the first medical device system; receiving a second message corresponding to a second request for an address from a second medical device system; determining that the second medical device system has replaced the first medical device system; changing an association of the first connection identifier from the first medical device system to the second medical device system; transmitting the first connection identifier to the second medical device system, wherein the second medical device system is further configured to update its identity with the first connection identifier.
12. The method of claim 11, further comprising facilitating transmission of a second message from the first medical device system to an additional hardware, wherein the first medical device system and the additional hardware are not directly connected.
13. The method of claim 11, further comprising transmitting messages between hardware components by copying a pointer and without copying the messages.
14. The method of claim 11, wherein a session layer communication is made independent of bus topology.
15. The method of claim 11, wherein a fragmented message is reassembled into a complete message in an application buffer.
16. The method of claim 11, further comprising determining that a size of data to transfer is larger than a predetermined fragmentation value and packing said data in a plurality of messages independent of an underlying data bus width.
17. The method of claim 11, further comprising requesting memory from a buffer comprising non-uniform sizes.