Institute of Biomedical Engineering
R.N. Scott Hall, 25 Dineen Drive
University of New Brunswick
Fredericton, NB E3B5A3 Canada

DRAFT COPY

Prosthetic Device Communication Protocol
for the
AIF UNB Hand Project

by

Yves Losier
UNB Hand Project Systems Integrator

File: IBME-PDCP (version 0_6_4).doc
2012-02-21

CONTENTS

1. DOCUMENT OVERVIEW

1.1. Purpose

This document gives an overview of the communication protocol between the different devices found within the AIF UNB Hand Project system. The Institute of Biomedical Engineering (IBME) has always had the intention of presenting the development of this protocol to the Standardised Communication Interface for Prosthetics (SCIP) "community" in the hopes that it could further advance the initiative's goal of developing an open, unified standard for the communication and power supply aspects of the interconnection of artificial limb components. This revised document by IBME is another step in fulfilling its intended contribution.

1.2. Scope

This document applies to the communication traffic between the Bus Arbitrator device and any other devices found within the prosthetic limb bus system. The hardware communications platform chosen is the CAN differential bus running in the standard identification setting at a rate of 1Mbps.

For additional bus communication references, a nice summary of various design consideration for the prosthetic field was written by Dr. Adrian Poulton and added to the Open Prosthetics Project website[1].

[1] http://openprosthetics.wikispot.org/Open_Standards_for_Prosthetics

2. CAN MESSAGE

2.1. Overview

A CAN message comprises of several different fields (Figure 2.1). Many CAN controller devices will automatically manage the contents of several of these fields, often termed module-controlled fields, such that the microprocessor unit only needs to be concerned with the user-controlled fields (Arbitration, Control, and Data fields).

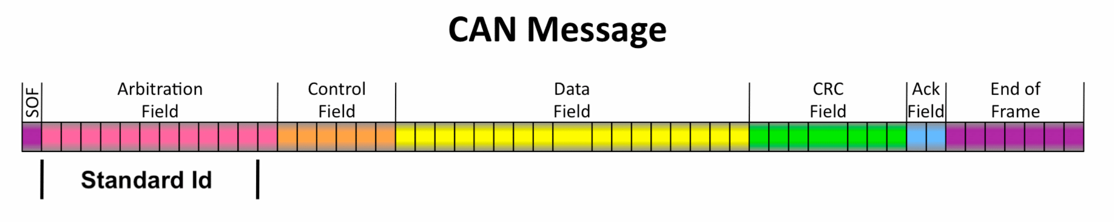


Figure 2.1 – CAN Message Standard Data Frame

2.2. Standard Identifier Field

The standard identifier (SID) is found within the arbitration field of the CAN message and is used to determine the message's priority on the communication bus. It is also used, in part or as a whole, for the message filtering process at each CAN controller.

For the purpose of this protocol, the standard identifier field is further divided into 3 subsections (Message Priority, Message Mode, and Node Identifier fields). These subsections are described in further detail below. The complete Standard Identifier Field map is illustrated in Figure 2.3.

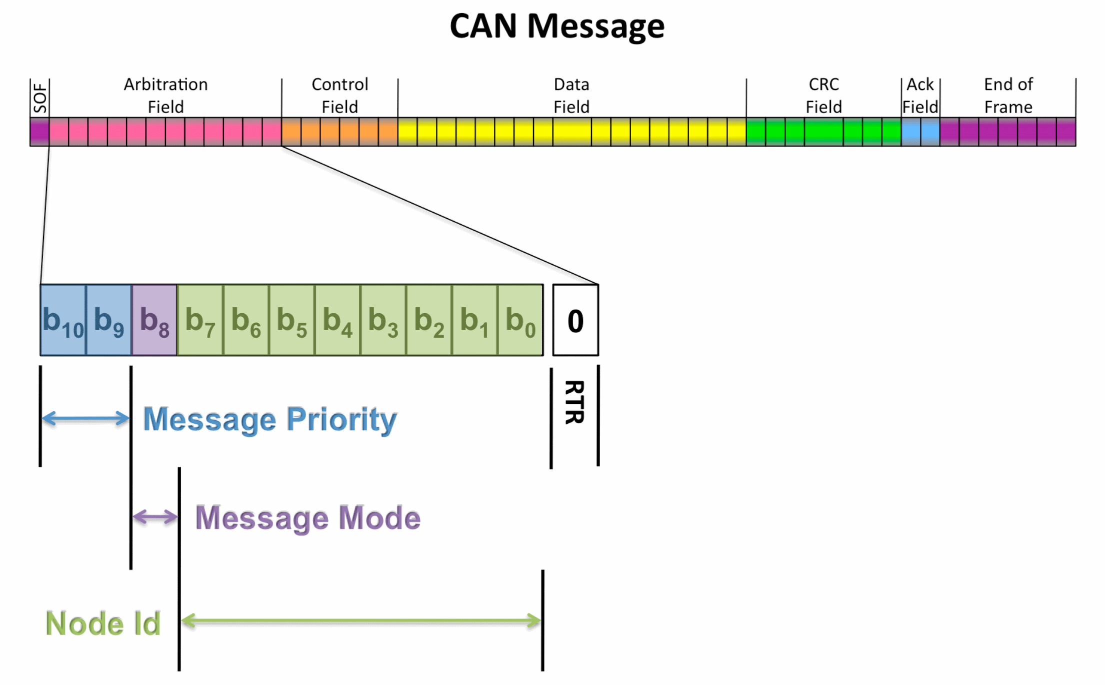


Figure 2.2 –Standard Identifier Field Subsections

2.2.1. Message Priority Field

The Message Priority field, as its name implies, is used to assign a priority to an outgoing CAN message. Although four possible values could be assigned to this field, only three are used during normal operation (0 ≡ High Priority, 1 ≡ Normal Priority, 2 ≡ Low Priority). A device, when attempting to bind itself to the bus system, uses the fourth value (3 ≡ Binding). If two or more devices attempt to send a message at the same time, the arbitration logic found within their CAN controllers will give bus control to the device whose message has a higher priority value. If two messages have the same priority level, the message arbitration continues into the Message Mode field (described below).

2.2.2. Message Mode Field

The Message Mode field is used to indicate whether a message is originating from the Bus Arbitrator (1 ≡ Bus Arbitrator Message Mode) or some other device (0 ≡ Standard Message Mode). If devices on the network are attempting to simultaneously transmit two or more messages (with identical Message Priority field values), the arbitration logic found within their CAN controllers will give bus control to the device whose message is using the Standard Message Mode. If two messages have the same priority level and message mode value, the message arbitration continues into the Node Identifier field (described below).

2.2.3. Node Identifier Field

The use of the Node Identifier field varies depending on the Message Mode field value. If a message originates from the Bus Arbitrator (i.e. Message Mode = 1), the Node Identifier field is assigned the Node Id of the intended recipient device. If the message is being transmitted using the Standard Message Mode (i.e. Message Mode = 0), the Node Identifier field is assigned the Node Id of the transmitting device's channel. The Node Id value range is between 1 (0x01) and 254 (0xFE) while the Node Id value of 0 is reserved for broadcast messages to the entire bus system.

Controller Area Network Standard Identifier Field				
P		M	N	Message Type Description
1	1	0	Node Id	Bind Device Request
1	1	1	Node Id	Not Used By Protocol
x	x	1	0	Broadcast Message
0	0	0	Node Id	High Priority Device Message/Data Transmission
0	0	1	Node Id	High Priority Bus Arbitrator Message Transmission
0	1	0	Node Id	Normal Priority Device Message/Data Transmission
0	1	1	Node Id	Normal Priority Bus Arbitrator Message Transmission
1	0	0	Node Id	Low Priority Device Message/Data Transmission
1	0	1	Node Id	Low Priority Bus Arbitrator Message Transmission

P=Priority , M=Mode , N=Node Id

Figure 2.3 – CAN Message Standard Identifier Field Map for the PDCP Standard

2.3. Standard Identifier Field Examples

New updated examples needed…

2.4. Data Field

The data field found within a CAN message can be between 0 and 8 bytes. This field is used to include function codes, parameters, and desired data. The actual function/message protocol is described in Section 3.

3. MESSAGE PROTOCOL

For the most part, the message protocol implemented is what is often referred to as a request-response message exchange model. In such a paradigm, the message sender will expect to receive a response message unless:

- The function code explicitly does not expect a response message to be returned.

3.1. Function Codes

The following function code list details the function code value, type, size as well as the sender and recipient devices. (Note: Deprecated functions are shaded in grey.)

Function Code	Function Code Description	Message Size (Bytes)	Sender	Recipient	Response Function Code
0x01	Bind Device Request	7	Device	Bus Arbitrator	0x81
0x02	GetDeviceInfo	2	Bus Arbitrator	Device	0x82
0x03	Get Device Parameter	3	Bus Arbitrator	Device	0x83
0x04	Set Device Parameter	4-7	Bus Arbitrator	Device	0x84
0x05	IndGetDeviceParameter	4	Device	Bus Arbitrator	0x85
0x06	IndSetDeviceParameter	5-8	Device	Bus Arbitrator	0x86
0x07	SetNodeId	2	Bus Arbitrator	Device	0x87
0x08	Suspend Device	3	Bus Arbitrator	Device	0x88
0x09	Release Device	1	Bus Arbitrator	Device	0x89
0x0A	Device Beacon	1	D or BA	BA or D	None
0x0B	Reset Device	1	Bus Arbitrator	Device	0x8B
0x0C	Configure Get Bulk Data Transfer	5	Bus Arbitrator	Device	0x8C
0x0D	Configure Set Bulk Data Transfer	5	Bus Arbitrator	Device	0x8D
0x0E	Bulk Data Transfer	3-8	D or BA	BA or D	0x8E
0x0F	Update Data Channel	2	Device	Bus Arbitrator	0x8F
0x10 - 0x4F	Reserved for Future System Commands	N/A	N/A	N/A	N/A
0x50 - 0x7F	Reserved for Module-Specific Commands	N/A	N/A	N/A	N/A
0x81	Bind Device Request Response	8	Bus Arbitrator	Device	None
0x82	GetDeviceInfoAck	2-6	Device	Bus Arbitrator	None
0x83	Get Device Parameter Response	4-8	Device	Bus Arbitrator	None
0x84	Set Device Parameter Response	5-8	Device	Bus Arbitrator	None
0x85	IndGetDeviceParameterAck	4-7	Bus Arbitrator	Device	None
0x86	IndSetDeviceParameterAck	1	Bus Arbitrator	Device	None
0x87	SetNodeIdAck	1	Device	Bus Arbitrator	None
0x88	Suspend Device Response	1	Device	Bus Arbitrator	None
0x89	Release Device Response	1	Device	Bus Arbitrator	None
0x8B	Reset Device Response	1	Device	Bus Arbitrator	None
0x8C	Configure Get Bulk Data Transfer Response	6	Device	Bus Arbitrator	None
0x8D	Configure Set Bulk Data Transfer Response	6	Device	Bus Arbitrator	None
0x8E	Bulk Data Transfer Response	2	D or BA	BA or D	None
0x8F	Update Data Channel Response	3	Bus Arbitrator	Device	None
0x90 - 0xCF	Reserved for Future System Responses	N/A	N/A	N/A	N/A
0xD0 - 0xFE	Reserved for Device-Specific Responses	N/A	N/A	N/A	N/A

Figure 3.1 – Function Code List for the Prosthetic Device Communication Protocol

3.2. Function Contents

This section provides additional information for the functions listed in Section 3.1.

3.2.1. 0x01 - Bind Device Request / 0x81 - Bind Device Request Response

Description: This function is sent by a device immediately following power-on or software reset. The Bus Arbitrator responds to the request by returning an available NodeId that has not been allocated to another device. If the NodeId value is identical to the one used to send the Bind Device Request, the device has been successfully bound to the system. If the NodeId value differs, the device will need to send a new Bind Device Request command using the new NodeId.

Originating Device: Any device during power-on or software resets
Destination Device: Bus Arbitrator

Command Message Format:

<table>
<tr><th colspan="8"><b>Function Code 0x01 - Bind Device Request</b> (DLC = 7)</th></tr>
<tr><td>Data$_0$</td><td>Data$_1$</td><td>Data$_2$</td><td>Data$_3$</td><td>Data$_4$</td><td>Data$_5$</td><td>Data$_6$</td><td>Data$_7$</td></tr>
<tr><td>0x01</td><td colspan="2">Device Vendor ID</td><td colspan="2">Device Product ID</td><td colspan="2">Device Serial Number</td><td></td></tr>
</table>

Response Message Format:

<table>
<tr><th colspan="8"><b>Function Code 0x81 - Bind Device Request Response</b> (DLC = 8)</th></tr>
<tr><td>Data$_0$</td><td>Data$_1$</td><td>Data$_2$</td><td>Data$_3$</td><td>Data$_4$</td><td>Data$_5$</td><td>Data$_6$</td><td>Data$_7$</td></tr>
<tr><td>0x81</td><td>NodeId</td><td colspan="2">Device Vendor ID</td><td colspan="2">Device Product ID</td><td colspan="2">Device Serial Number</td></tr>
</table>

3.2.2. 0x03 - Get Device Parameter / 0x83 - Get Device Parameter Response

Description: This function is sent by the Bus Arbitrator to request a parameter value from the specified device.

Originating Device: Bus Arbitrator
Destination Device: Any Device

Command Message Format:

Function Code 0x03 - Get Device Parameter							(DLC = 3)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x03	Parameter Id	Channel Index					

Response Message Format:

Function Code 0x83 - Get Device Parameter Response							(DLC = 4-8)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x83	Response Code	Parameter Id	Channel Index	Parameter Value (0 - 4 bytes)			

Response Codes:

0. Failure Response
1. Successful Response
2. ‘Use Get Bulk Data Command’ Response

3.2.3. 0x04 - Set Device Parameter / 0x84 - Set Device Parameter Response

Description: This function is sent by the Bus Arbitrator to set a parameter value of the specified device.

Originating Device: Bus Arbitrator
Destination Device: Any Device

Command Message Format:

Function Code 0x04 - Set Device Parameter							(DLC = 4-7)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x04	Parameter Id	Channel Index	Parameter Value (1 - 4 bytes)				

Response Message Format:

Function Code 0x84 - Set Device Parameter Response							(DLC = 5-8)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x84	Response Code	Parameter Id	Channel Index	Parameter Value (1 - 4 bytes)			

Response Codes:

0. Failure Response
1. Successful Response
2. ‘Use Set Bulk Data Command’ Response

3.2.4. 0x08 - Suspend Device / 0x88 - Suspend Device Response

Description: This function is sent by the Bus Arbitrator to request that the specified device cease to transmit data on the bus for the specified period of time. The parameter, TimeValue, can be set to 0 if indefinite suspension of the device data transmission is desired.

Originating Device: Bus Arbitrator
Destination Device: Any Device

Command Message Format:

Function Code 0x08 - Suspend Device							(DLC = 3)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x08	TimeValue (msec)						

Response Message Format:

Function Code 0x88 - Suspend Device Response							(DLC = 4)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x88	Response Code	TimeValue (msec)					

Response Codes:

0. Failure Response
1. Successful Response

3.2.5. 0x09 - Release Device / 0x89 - Release Device Response

Description: This function is sent by the Bus Arbitrator to enable the specified device to transmit data on the bus.

Originating Device: Bus Arbitrator
Destination Device: Any Device

Command Message Format:

Function Code 0x09 - Release Device							(DLC = 1)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x09							

Response Message Format:

Function Code 0x89 - Release Device Response							(DLC = 2)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x89	Response Code						

Response Codes:

0. Failure Response
1. Successful Response

3.2.6. 0x0A - Device Beacon

Description: This function is sent by either the Bus Arbitrator or a device to inform others of its continued presence on the bus.

Originating Device: Any Device | Bus Arbitrator
Destination Device: Bus Arbitrator | Any Device

Command Message Format:

Function Code 0x0A - Device Beacon (DLC = 1)							
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x0A							

3.2.7. 0x0B - Reset Device / 0x8B - Reset Device Response

Description: This function is sent by the Bus Arbitrator to reset the specified device.

Originating Device: Bus Arbitrator
Destination Device: Any Device

Command Message Format:

Function Code 0x0B - Reset Device							(DLC = 1)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x0B							

Response Message Format:

Function Code 0x8B - Reset Device Response							(DLC = 2)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x8B	Response Code						

Response Codes:

0. Failure Response
1. Successful Response

3.2.8. 0x0C - Configure Get Bulk Data Transfer / 0x8C - Configure Get Bulk Data Transfer Response

Description: This function is sent by the Bus Arbitrator to acquire a large amount of data for a given parameter of the specified device.

Originating Device: Bus Arbitrator
Destination Device: Any Device

Command Message Format:

Function Code 0x0C – Configure Get Bulk Data Transfer (DLC = 3)							
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x0C	Parameter Id	Channel Index					

Response Message Format:

Function Code 0x8C – Configure Get Bulk Data Transfer Response (DLC = 6)							
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x8C	Response Code	Parameter Id	Channel Index	Bulk Data Size			

Response Codes:

0. Failure Response
1. Successful Response

3.2.9. 0x0D - Configure Set Bulk Data Transfer / 0x8D - Configure Set Bulk Data Transfer Response

Description: This function is sent by the Bus Arbitrator to initiate the transfer of a large amount of data for the given parameter of the specified device.

Originating Device: Bus Arbitrator
Destination Device: Any Device

Command Message Format:

Function Code 0x0D – Configure Set Bulk Data Transfer							(DLC = 5)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x0D	Parameter Id	Channel Index	Bulk Data Size				

Response Message Format:

Function Code 0x8D – Configure Set Bulk Data Transfer Response							(DLC = 6)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x8D	Response Code	Parameter Id	Channel Index	Bulk Data Size			

Response Codes:

0. Failure Response
1. Successful Response

3.2.10. 0x0E - Bulk Data Transfer / 0x8E - Bulk Data Transfer Response

Description: This function is used by either the Bus Arbitrator or the specified device in order to transfer a large amount of data.

Originating Device: Bus Arbitrator | Any Device
Destination Device: Any Device | Bus Arbitrator

Command Message Format:

Function Code 0x0E - Bulk Data Transfer							(DLC = 3-8)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x0E	Packet Id	Bulk Data (1-6 bytes)					

Response Message Format:

Function Code 0x8E - Bulk Data Transfer Response							(DLC = 2)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x8E	Response Code						

Response Codes:

0. Failure Response
1. Successful Response

3.2.11. 0x0F - Update Data Channel / 0x8F - Update Data Channel Response

Description: This function is sent by a device requesting that the Bus Arbitrator updates the nodeIds corresponding to the data streaming source for a device's input data channel.

Originating Device: Any Device
Destination Device: Bus Arbitrator

Command Message Format:

Function Code 0x0F - Update Data Channel							(DLC = 2)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x0F	Channel Index						

Response Message Format:

Function Code 0x8F - Update Data Channel Response							(DLC = 3)
Data$_0$	Data$_1$	Data$_2$	Data$_3$	Data$_4$	Data$_5$	Data$_6$	Data$_7$
0x8F	Response Code	Channel Index					

Response Codes:

0. Failure Response
1. Successful Response

3.3. Function Code Implementation Examples

This section illustrates examples of the function codes described in Sections 3.1 and 3.2.

3.3.1. Device Binding

SID			DLC	Data								Message Description
Priority	Message Mode	Node Id		Data_0	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	
3	0	1	7	0x01	0x01	0x00	0x0A	0x00	0x04	0x00		Bind Request from Device (VID = 0x0001, PID = 0x000A, SN = 0x0004)
1	1	1	8	0x81	0x05	0x01	0x00	0x0A	0x00	0x04	0x00	Bind Request Response requesting the Device retries with nodeId 0x05
3	0	5	7	0x01	0x01	0x00	0x0A	0x00	0x04	0x00		Bind Request from Device (VID = 0x0001, PID = 0x000A, SN = 0x0004)
1	1	5	8	0x81	0x05	0x01	0x00	0x0A	0x00	0x04	0x00	Bind Request Response acknowledging the Device request for nodeId = 0x05

3.3.2. Get Device Parameter

SID			DLC	Data								Message Description
Priority	Message Mode	Node Id		Data_0	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	
1	1	5	3	0x03	0x09	0x00						Get Device Parameter (Number of Data Channels)
1	0	5	5	0x83	0x01	0x09	0x00	0x01				Get Device Parameter Resonse (Number of Data Channels = 1)

3.3.3. Set Device Parameter

SID			DLC	Data								Message Description
Priority	Message Mode	Node Id		Data_0	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	
1	1	5	4	0x04	0x05	0x01	0x06					Set Device Parameter (Data Channel #1's Node Id = 0x06)
1	0	5	5	0x84	0x01	0x05	0x01	0x06				Set Device Parameter Resonse (Data Channel #1's Node Id = 0x06)

3.3.4. Device Beacon

SID			DLC	Data								Message Description
Priority	Message Mode	Node Id		Data_0	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	
0	0	5	1	0x0A								Node Beacon from Device (VID = 0x0001, PID = 0x000A, SN = 0x0004)

3.3.5. Reset Device

SID			DLC	Data								Message Description
Priority	Message Mode	Node Id		Data_0	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	
1	1	5	1	0x0B								Reset Device
1	0	5	2	0x8B	0x01							Reset Device Response (Acknowledging Reset Request)

3.3.6. Get Device Parameter, Configure Get Bulk Data Transfer & Bulk Data Transfer

SID			DLC	Data								Message Description
Priority	Message Mode	Node Id		Data_0	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	
1	1	5	3	0x03	0x07	0x00						Get Device Parameter (Device Descriptor)
1	0	5	4	0x83	0x02	0x07	0x00					Get Device Parameter Response (Response Code: Use Get Bulk Data Command)
1	1	5	3	0x0C	0x07	0x00						Configure Get Bulk Data Transfer (Device Descriptor)
1	0	5	6	0x8C	0x01	0x07	0x00	0x21	0x00			Configure Get Bulk Data Transfer Response (Device Descriptor, Size: 33 bytes)
1	0	5	8	0x0E	0x01	0x50	0x72	0x6F	0x74	0x6F	0x74	Bulk Data Transfer (Packet Id: 1)
1	0	5	8	0x0E	0x02	0x79	0x70	0x65	0x20	0x50	0x44	Bulk Data Transfer (Packet Id: 2)
1	0	5	8	0x0E	0x03	0x43	0x50	0x20	0x44	0x65	0x76	Bulk Data Transfer (Packet Id: 3)
1	0	5	8	0x0E	0x04	0x69	0x63	0x65	0x20	0x44	0x65	Bulk Data Transfer (Packet Id: 4)
1	0	5	8	0x0E	0x05	0x73	0x63	0x72	0x69	0x70	0x74	Bulk Data Transfer (Packet Id: 5)
1	0	5	5	0x0E	0x06	0x6F	0x72	0x00				Bulk Data Transfer (Packet Id: 6)
1	1	5	2	0x8E	0x01							Bulk Data Transfer Response

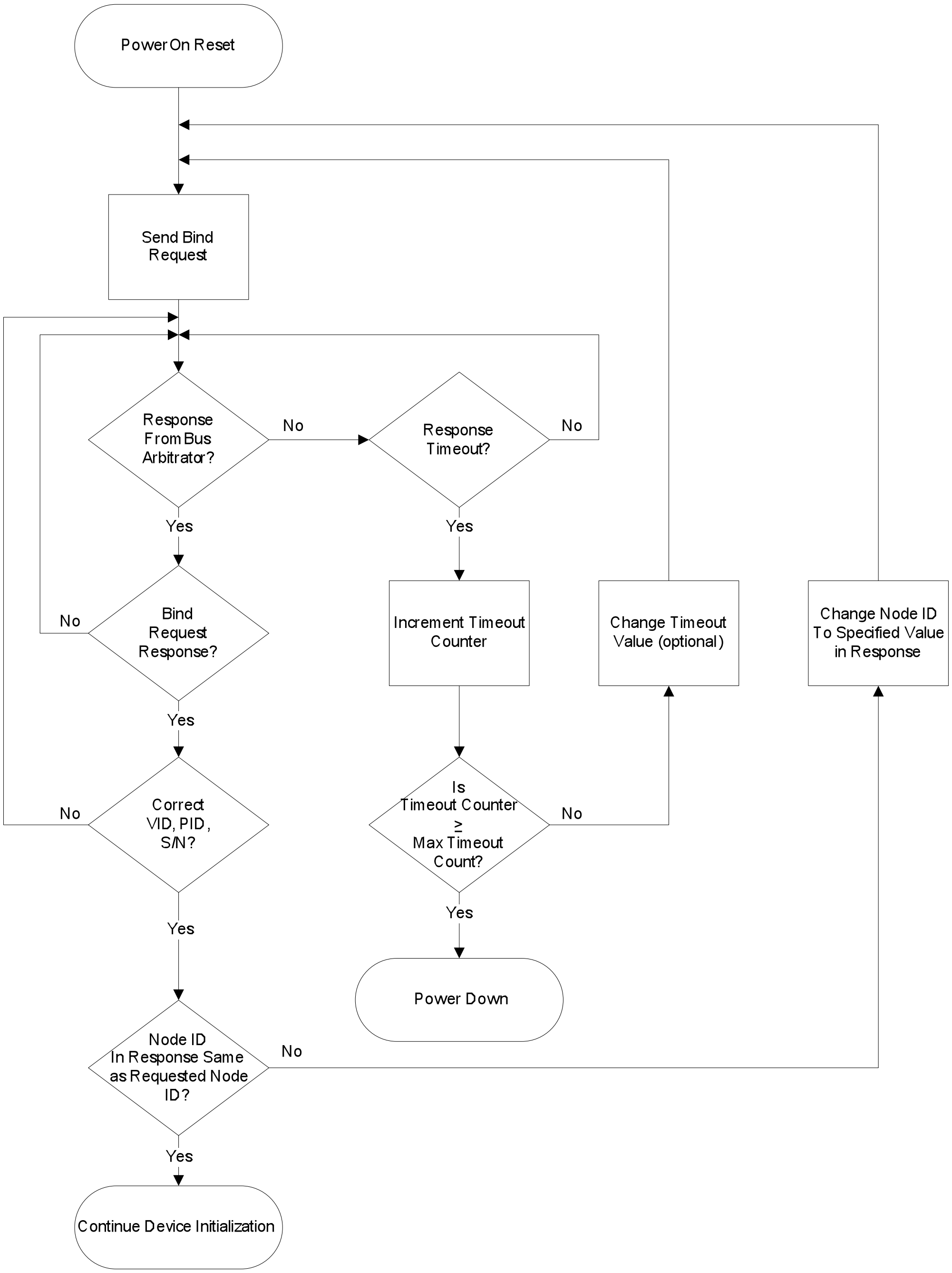
3.3.7. Configure Set Bulk Data Transfer & Bulk Data Transfer

SID			DLC	Data								Message Description
Priority	Message Mode	Node Id		Data_0	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	
1	1	5	5	0x0D	0x37	0x00	0x11	0x00				Configure Set Bulk Data Transfer (Arbitrary Parameter for this example,Size: 17 bytes)
1	0	5	6	0x8D	0x01	0x37	0x00	0x11	0x00			Configure Set Bulk Data Transfer Response
1	1	5	8	0x0E	0x01	0x01	0x02	0x03	0x04	0x05	0x06	Bulk Data Transfer (Packet Id: 1)
1	1	5	8	0x0E	0x02	0x07	0x08	0x09	0x0A	0x0B	0x0C	Bulk Data Transfer (Packet Id: 2)
1	1	5	7	0x0E	0x03	0x0D	0x0E	0x0F	0x10	0x11		Bulk Data Transfer (Packet Id: 3)
1	0	5	2	0x8E	0x01							Bulk Data Transfer Response

3.3.8. Update Data Channel

SID			DLC	Data								Message Description
Priority	Message Mode	Node Id		Data_0	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	
1	0	5	2	0x0F	0x04							Update Data Channel (Data Channel #4)
1	1	5	3	0x8F	0x01	0x04						Update Data Channel Response (Data Channel #4)

Bind Request Flow Chart (Device)


Power On Reset
Send Bind Request
Response From Bus Arbitrator?
No
Response Timeout?
No
Yes
Yes
Bind Request Response?
No
Increment Timeout Counter
Change Timeout Value (optional)
Change Node ID To Specified Value in Response
Yes
Correct VID, PID, S/N?
No
Is Timeout Counter ≥ Max Timeout Count?
No
Yes
Yes
Power Down
Node ID In Response Same as Requested Node ID?
No
Yes
Continue Device Initialization

3.4. Device Profile Layer

The protocol will require standardised device/channel types and profiles to facilitate interchangeability of devices. Channel type and profile define the characteristics of the streaming data. Although none of these have been mentioned or specifically addressed in this document, efforts were made not to inhibit its potential implementation at a later date.

4. REVISION HISTORY

Version	Date	Author	Description
0.1	3-Sep-09	Y. Losier	- Preliminary draft circulated to IBME staff for feedback.
0.2	14-Sep-09	Y. Losier	- Modified document based on feedback from Adam Wilson.
0.3	15-Sep-09	Y. Losier	- Modified document based on feedback from Øyvind Stavdahl.
0.4	21-Sep-09	Y. Losier	- Added additional standard identifier field examples to illustrate communication in a multi-Bus Arbitrator system and broadcast messaging.
0.5	JAN-10	Y. Losier	- Added additional functional code examples - Modified functional code structure based on suggestion from RIC
0.6.0	July-11	Y. Losier	- Updated protocol documentation to reflect current implementation of PDCP
0.6.1 to 0.6.4	Aug-11 to Feb-12	Y. Losier	- Updating documentation with improved diagrams, flowcharts, and explanations.