****

**Rochester Hills**

Design

RP1210 and J1939 Primer

|  |  |
| --- | --- |
| Revision # | 2 |
| Revision Date | 10/11/17 |

Proprietary & Confidential

Design

RP1210 and J1939 Primer

Written by: Todd B Norris

Table of Contents

1.0 Requirements 1

2.0 RP1210 Connection 1

2.1 Determine Device to be used 1

2.1.1 Read RP121032.ini 1

2.1.2 RP1210\_Ioctl() 2

2.2 Life of an RP1210 Connection 2

2.2.1 RP1210\_ClientConnect(…) 2

2.2.1.1 Client 2

2.2.1.2 DeviceID 2

2.2.1.3 Protocol String 2

2.2.1.4 TxBufferSize and RcvBufferSize 2

2.2.1.5 IsAppPacketizingIncomingMsgs 2

2.2.1.6 Return Code 3

2.2.2 Starting the Message Pump 3

2.2.2.1 RP1210\_SendCommand 3

2.2.2.1.1 Setting Echo (Command 16) 3

2.2.2.1.2 All Pass Filter (Command 3) 3

2.2.2.1.3 Any other filtering 3

2.2.3 Setup Read Thread 3

2.2.3.1 RP1210\_ReadMessage 3

2.2.3.1.1 ClientID 3

2.2.3.1.2 RxMessage 4

2.2.3.1.3 RxMessageLength 4

2.2.3.1.4 IsBlocking 4

2.2.3.2 Process the message 4

2.2.4 Writing messages to the bus 4

2.2.4.1 Claim an Address (J1939 Only) 4

2.2.4.1.1 RP1210\_SendCommand 4

2.2.4.2 RP1210\_SendMessage 4

2.2.4.2.1 ClientID 4

2.2.4.2.2 TxMessage 4

2.2.4.2.3 TxMessageLength 5

2.2.4.2.4 NotifyStatusOnTx: 5

2.2.4.2.5 IsBlocking 5

2.2.4.2.6 Return code 5

2.2.5 Closing down the Connection 5

2.2.5.1 Stop any SendMessage calls 5

2.2.5.2 Kill your Read Thread 5

2.2.5.3 RP1210\_ClientDisconnect 5

2.2.5.3.1 ClientID 5

3.0 J1939 Primer 5

3.1 J1939 Message structure 5

3.1.1 J1939 Header 6

3.1.1.1 PGN 6

3.1.1.2 HOW/Priority Byte 6

3.1.1.3 Source Address 6

3.1.1.4 Destination Address 6

3.1.2 J1939 Data Packet 6

3.2 J1939 Practical Example 6

3.2.1 Making a request of the bus 7

3.2.1.1 Code example 7

3.2.2 Retrieving messages from the bus 7

3.2.2.1 PGN Response 7

3.2.2.2 ACK Response 8

3.2.2.3 Code Example 8

3.2.3 Final J1939 Notes 8

Change Log 9

1. Requirements

In order to use get a complete understanding of RP1210 you need to acquire the latest RP 1210C document from TMC.

* RP1210C: <http://www.atabusinesssolutions.com/ATAStore/ProductDetails?productid=2675472>

For J1939, the following documents are helpful to understand the protocol:

* J1939/21: Data Link Layer, messaging definitions, including how to setup a request message and how the response message should look <http://standards.sae.org/j1939/21_201603/>
* J1939/DA: Data Annex, contains PGN/SPN descriptions and data breakdowns <http://standards.sae.org/j1939da_201707/>
* J1939/71: Vehicle Application Layer, contains information on parameter range definitions, transmission rates, parameter naming conventions, and ASCII usage definitions, <http://standards.sae.org/j1939/71_201610/>
* J1939/73: Diagnostics Application Layer, contains all the necessary diagnostic PGNs defined by regulatory agencies, <http://standards.sae.org/j1939/73_201705/>

1. RP1210 Connection
   1. Determine Device to be used

There are two ways to determine the device to be used.

* + 1. Read RP121032.ini

This file lists all RP1210 implementations installed on the computer and is located in C:\Windows.

The INI file has each implementation listed in a comma separated list under the Header [RP1210Support] key “APIImplementations.” Each implementation will have a file that is <implementation name>.ini (e.g. NULN2R32.ini)

Device IDs are located under the implementation’s [DeviceInformation{X}] under the key “DeviceID”

Protocols supported by the implementation are listed under the header [ProtocolInformation{X}]. The following keys: ProtocolSpeed, ProtocolString, and Devices are all needed for connection over RP1210.

Also listed in the INI is the connection string format for each protocol supported by the implementation. The comments listed above each “FormatsSupported” shows the connection strings, i.e. for J1939 the format 1 comment shows "J1939:Baud=X" which means J1939 protocol with a Baud rate of X, where X is Auto, 250, 500, 1000.

* + 1. RP1210\_Ioctl()

Use the IOCTL\_DISCOVERY command.

* 1. Life of an RP1210 Connection

The following sections show a typical RP1210 connection. The main parts of a connection are connect, set filters, set up a read thread, write messages to the bus, and when complete close the connection.

* + 1. RP1210\_ClientConnect(…)

**Prototype:** RP1210\_ClientConnect(Client, DeviceID, Protocol String, TxBufferSize, RcvBufferSize, IsAppPacketizingIncomingMsgs)

* + - 1. Client

This is unneeded now, so set it to 0x00.

* + - 1. DeviceID

This is found in <implementation name>.ini [DeviceInformation{X}]->DeviceID

* + - 1. Protocol String

Made up of a few parts in RP1210C, example “J1939:Baud=Auto, Channel=1,DiscoveryString=00:07:80:0D:6A:96”

For J1939,the defaults (unspecified) are Baud=250 and Channel=1, if you are using the ini method, you also don’t need to use the DiscoveryString section

* **ProtocolName:** <implementation name>.ini->[ProtocolInformation{X}]->ProtocolString
* **Baud:** Only really needed in cases that there is more than one speed listed, but “Auto” is a good option for simplification.
* **Channel:** There can be more than one channel usually 1 is used
* **DiscoveryString:** (MAC Address from using the Ioctl method)
  + - 1. TxBufferSize and RcvBufferSize

Should be large enough to account for the queue getting backed up, 1 m is typically a good size for both.

* + - 1. IsAppPacketizingIncomingMsgs

Unless there is a specific reason to use this, the value should be set to 0. There is very little reason that the VCI driver/firmware should not be handling this on the consumer’s behalf.

* + - 1. Return Code

This is an error if it is 128 or greater, if it is under 128 it is your Client ID. This value is used for most other RP1210 call so it must be saved for the life of the connection.

* + 1. Starting the Message Pump

There are two commands that need to be set to get the basics start up messaging to properly work in mobile platforms, Set Echo and Pass All Filter.

* + - 1. RP1210\_SendCommand

**Prototype:** RP1210\_SendCommand(Command, ClientID, Data, DataLength)

* + - * 1. Setting Echo (Command 16)
* **Command:** 16
* **ClientID:** What was returned from RP1210\_ClientConnect
* **Data:** An array with the value { 1 }
* **DataLength:** 1
  + - * 1. All Pass Filter (Command 3)
* **Command:** 3
* **ClientID:** What was returned from RP1210\_ClientConnect
* **Data:** NULL
* **DataLength:** 0
  + - * 1. Any other filtering

Now any other filtering can be applied from here until we close the connection

* + 1. Setup Read Thread

In order to not have the message queue full error, the read thread must run fast enough to pull messages out of the queue. This thread needs to be as quick as possible to ensure timely message processing.

* + - 1. RP1210\_ReadMessage

Pulls messages off of the queue for processing by the client.

**Prototype:** RP1210\_ReadMessage(ClientID, RxMessage, RxMessageLength, IsBlocking)

* + - * 1. ClientID

The identifying number that was given during the RP1210\_ClientConnect.

* + - * 1. RxMessage

Should be an array large enough to place the largest message for the protocol.

* + - * 1. RxMessageLength

How much space has the driver been given, and this will also report back how big of a message was placed in RxMessage.

* + - * 1. IsBlocking

Value can be either NON\_BLOCKING\_IO or BLOCKING\_IO. It is recommended to use BLOCKING\_IO so that the VCI does not need to respond to the caller until a message is available.

* + - 1. Process the message

Now that the message have been received it can be acted upon.

* + 1. Writing messages to the bus
       1. Claim an Address (J1939 Only)

This only needs to be done once per client, and no address can be claimed by two bus clients.

* + - * 1. RP1210\_SendCommand
* **Command:** 19
* **ClientID:** What was returned from RP1210\_ClientConnect
* **Data:** 10 Bytes
* Byte[0]: Source Address being claimed
* Byte[1-8]: Network Name

Byte[9]: Status Byte (0 or 2)

* **Return Code:** 0 is success
  + - 1. RP1210\_SendMessage

This is the call used to place a message on the bus.

**Prototype:** RP1210\_SendMessage(ClientID, TxMessage, TxMessageLength, NotifyStatusOnTx, IsBlocking)

* + - * 1. ClientID

The ID that was returned from the RP1210\_ClientConnect call.

* + - * 1. TxMessage

The array with the message to transmit.

* + - * 1. TxMessageLength

The length of the message.

* + - * 1. NotifyStatusOnTx:

Unused, so use 0x00.

* + - * 1. IsBlocking

Either NON\_BLOCKING\_IO (0) or BLOCKING\_IO (1)

* + - * 1. Return code

0x00 is success, and any return value > 127 is an error, see RP1210C section 23.10 for errors.

* + 1. Closing down the Connection

When the connection is no longer needed, it must be closed down to free up the VCI for new connections.

* + - 1. Stop any SendMessage calls

Shutdown your requests.

* + - 1. Kill your Read Thread

Stop reading data from the VCI to it can shut down. An error will also be returned from RP1210\_ReadMessage after RP1210\_ClientDisconnect is called.

* + - 1. RP1210\_ClientDisconnect

Prototype: RP1210\_ClientDisconnect(Client ID)

* + - * 1. ClientID

The client ID returned from the RP1210\_ClientConnect command.

1. J1939 Primer

This section goes over the basics of sending and receiving J1939 messages. Message parts that span more than one byte are in little endian, except for ASCII fields.

* 1. J1939 Message structure

The J1939 Message breaks down as follows:

[PGN0] [PGN1] [PGN2] [How/Priority] [Source Address] [Destination Address] [Data 0..7]

There are two basic parts of the message, the first contains the header information and the second part is the message data.

* + 1. J1939 Header

The header consists of 6 bytes. There are 4 major parts in the header.

* + - 1. PGN

This is the first 3 bytes of any message.

Using an example of ECU History PGN 65201 (0x00FEB1)

Message PGN: B1 FE 00

* + - 1. HOW/Priority Byte

This is the 4th byte of the message.

The highest bit (0x80) is the HOW byte, unless there is a specific reason leave this bit low.

The rest of this byte is priority, in almost all cases the priority should be 6.

Message HOW/Priority: 0x06

* + - 1. Source Address

This comes from the J1939/DA, the most common SAs are Engine#1 (0x00) and Off-board Diagnostic Tool (0xF9). This is the address that the message came FROM. Acceptable values are 0x00-0xFD for Source Address.

* + - 1. Destination Address

This also comes from J1939/DA, it usage is similar to Source Address. The difference is this is who the message is directed towards. The only other difference is a Global Address (0xFF) can be used. When used as a global request a response is not guaranteed, but if the controller supports it and it is able to send the data, it will.

* + 1. J1939 Data Packet

The rest of the message starting with the 7th byte of the message is all data, and its handling completely depends on the PGN.

As a general rule, all data packets are 8 bytes or longer, any message longer than 8 bytes must be packetized by the VCI and thus will take slightly more time for the other end to get the fully assembled message. The only real exception to this is the Request PGN (0x00EA00) its data packet is just 3 bytes, the three bytes of the PGN that is being requested.

* 1. J1939 Practical Example

In this practical example, we are trying to part Total Vehicle Distance (SPN 1032) from the Engine (Source Address 0x00) as an Off-board Diagnostic Tool (Source Address 0xF9).

* + 1. Making a request of the bus

According to the J1939/DA, this SPN is sent over the PGN call “ECU History” (0x00FEB1) which is “On Request.” This means we must use the Request PGN (0x00EA00).

* + - 1. Code example

byte message[] = { 0x00, 0xEA, 0x00, 0x06, 0xF9, 0x00, 0xB1, 0xFE, 0x00 };

short messageLength = \_countof(message);

short return = RP1210\_SendMessage(0, clientId, message, messageLength, 0, 0);

* + 1. Retrieving messages from the bus

This is when you message thread come into play. You can get 2 different kinds of responses based on your request.

* + - 1. PGN Response

This is the best case, the controller can respond and does. This will look like the following:

B1 FE 00 07 00 FF D9 40 18 00 D1 01 00 00

Notice that the Destination Address is not 0xF9, if a message data packet is 8 bytes (or less) a global destination will sometimes be used, this is normal and acceptable.

Looking at J1939/DA we see two data points under PGN 65201 (0x00FEB1) SPN 1032 Total ECU Distance and SPN 1033 Total ECU Time

The Data packet is: D9 40 18 00 D1 01 00 00

SPN 1032 is in position 1-4 (one’s based) or: D9 40 18 00

* **Resolution:** 0.125 km/bit and an offset of 0
* **Raw Data:** 0x001840D9 or 1589465
* **Apply scaling:** 1589465 bit \* 0.125 km/bit (-offset of 0) = 198683.125 km
* **For English units:** 198683.125 km \* 0.621371 mi/km = 123455.9702587 mi or 123456 mi

SPN 1033 states that it is at Position 5-8 (one’s based) or D1 01 00 00

* **Resolution:** 0.05 hr/bit with an offset of 0
* **Raw Data:** 0x000001D1 or 465
* **Apply scaling:** 465 \* 0.05 hr/bit (- offset of 0) = 23.25 hours.
  + - 1. ACK Response

If a global request is made, an ACK is never sent, this is to cut down on bus traffic.

The following is an example of an ACK PGN send as a NACK

00 E8 00 06 00 FF 01 FF FF FF F9 B1 FE 00

* **PGN:** 0x00E800 (ACK PGN)
* **Source Address:** 0x00 (Engine)
* **Data packet:** 01 FF FF FF F9 B1 FE 00

1st Byte is the ACK Reason, in this case it is 0x01, and normal values are:

* 0x00 : ACK message (could be send back for a fault clear request message)
* 0x01 : NACK (controller does not support that message)
* 0x02 : Access Denied
* 0x03 : Cannot Respond, try again later

Byte 5: is the requestor’s address

Byte 6-8: is the PGN the requestor was requesting

* + - 1. Code Example

void Read Thread()

{

while (canRun)

{

byte message[4096]

ZeroMemory(message, sizeof(message));

byte messageLength = \_countof(message);

short return = RP1210\_ReadMessage(clientId, message, messageLength, 0);

//Look at message

}

* + 1. Final J1939 Notes

If the message is marked in the J1939/DA as having an interval, then it MUST NOT be requested, just continue to call RP1210\_ReadMessage, until the needed PGN is received.

Change Log

| Rev | 1. Author | 1. Sections Changed | 1. Change Description | Date |
| --- | --- | --- | --- | --- |
|  | Todd Norris | All sections | Initial creation. | 10/09/17 |
|  | Jeff Meller | All sections | Minor Updates | 10/11/17 |

*(To update all dates and Table of Contents, press Ctrl + a, then press F9.)*

(Current revision 2; last revision date: 10/11/17 Please do not disturb this line of text; it supports automatic version numbering.)

This document is archived in SVN under: $Insert VSS location.