Android Library

Table of Contents

1 Features Highlights

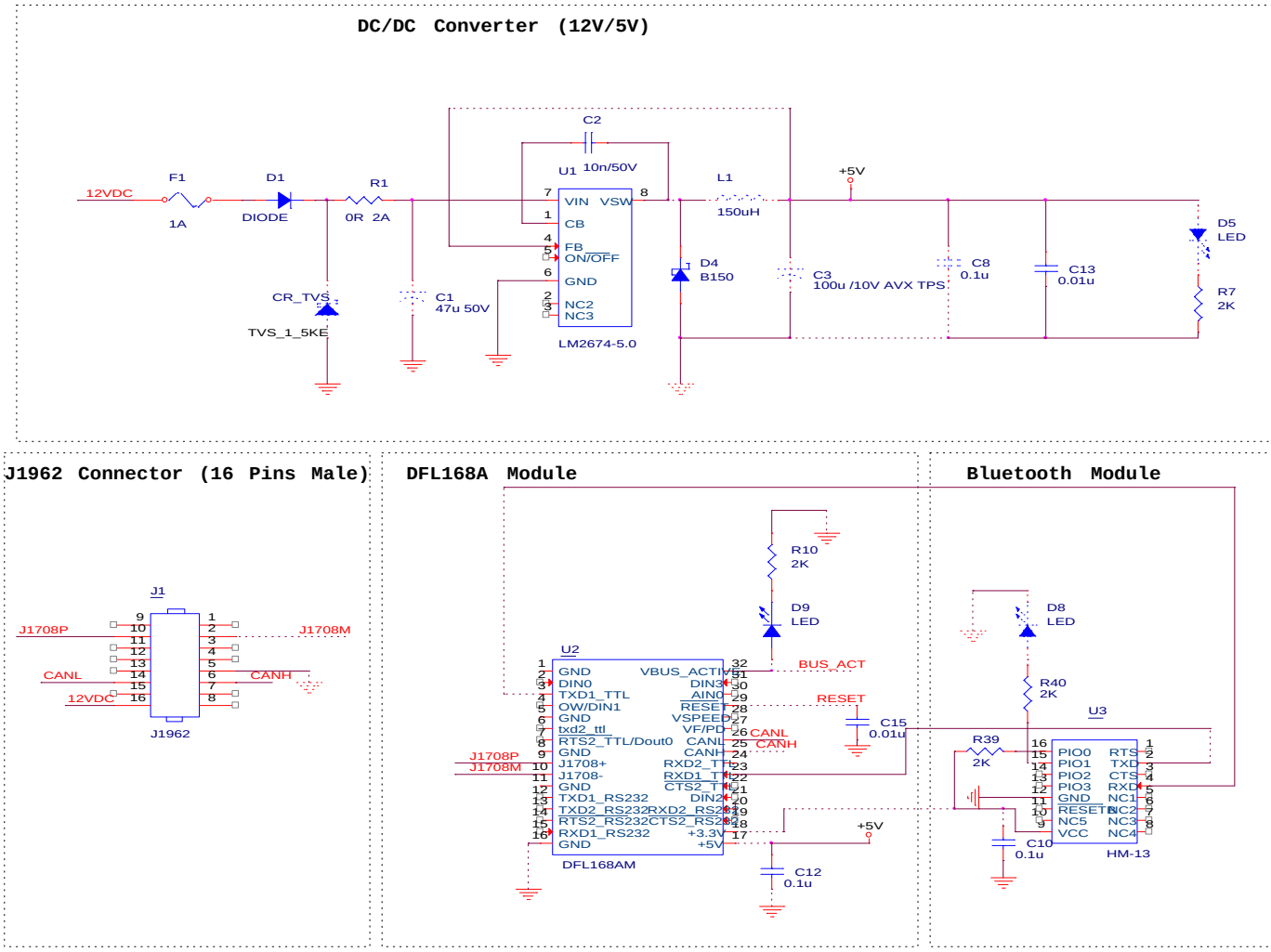
- Simply use our object DFL168A and its method to get motor data
- Don't need to know OBD2 and J1939/J1708/J1587 protocol, don't need to read DFL168A data sheet, just need to know DFL168A pinout, so easily get vehicle parameters values in real time.
- Can be used in Bluetooth BR/EDR (Classic Bluetooth) hardware environment
- Can use our android Bluetooth Device Selection/Scan Graphic interface, Or use yourself Bluetooth Device Selection/Scan Graphic interface.

2 Hardware

You must use any classic Bluetooth (BR/EDR) Slave Uart module such as HM-13,HM-17,RN-52, HC-05,HC-06,BM78

You cannot use BLE because our library only supports classic (BR/EDR) Bluetooth.

The Drawing of DFL168AM (Module) and HM-13 (Module) is shown below:



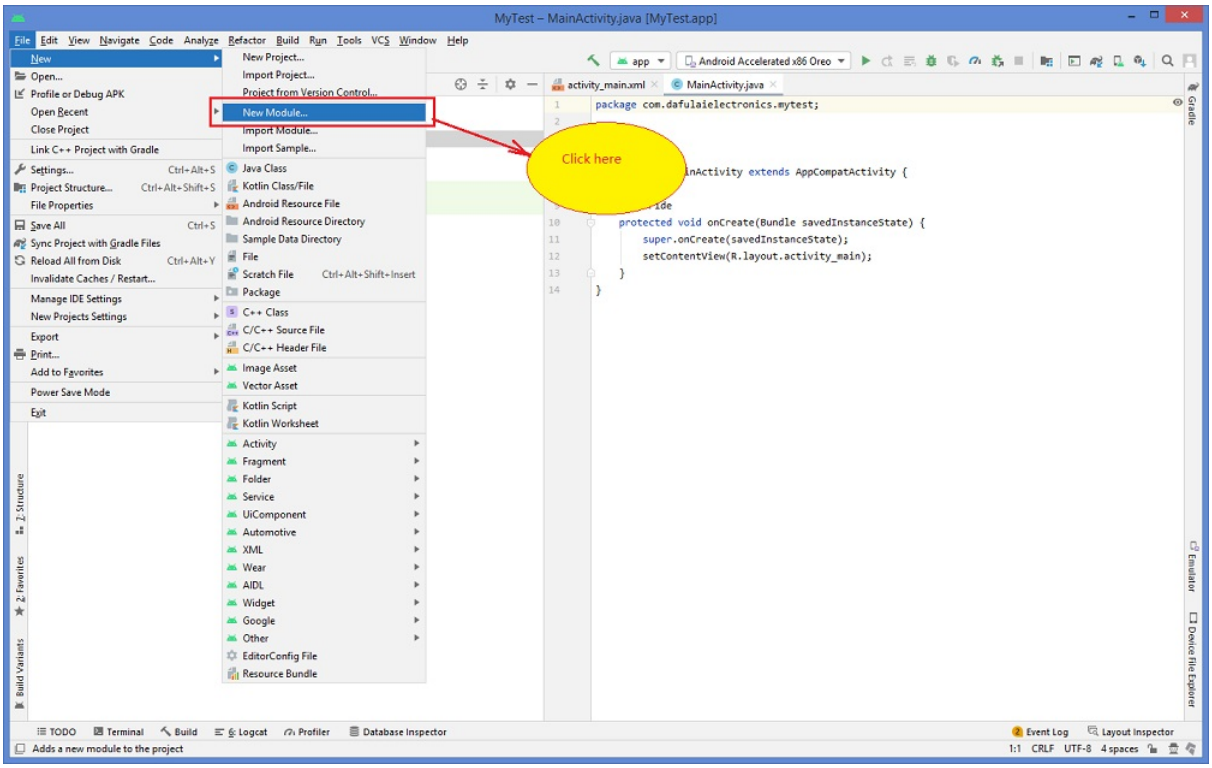
Notes: *If you use DFL168A IC, please read DFL168A reference schematic in page 8 of [DFL168A Datasheet](), And for the other Bluetooth modules, please read your specific Bluetooth module datasheet, You can make your customized schematic*

3 How to install library in Android Studio?

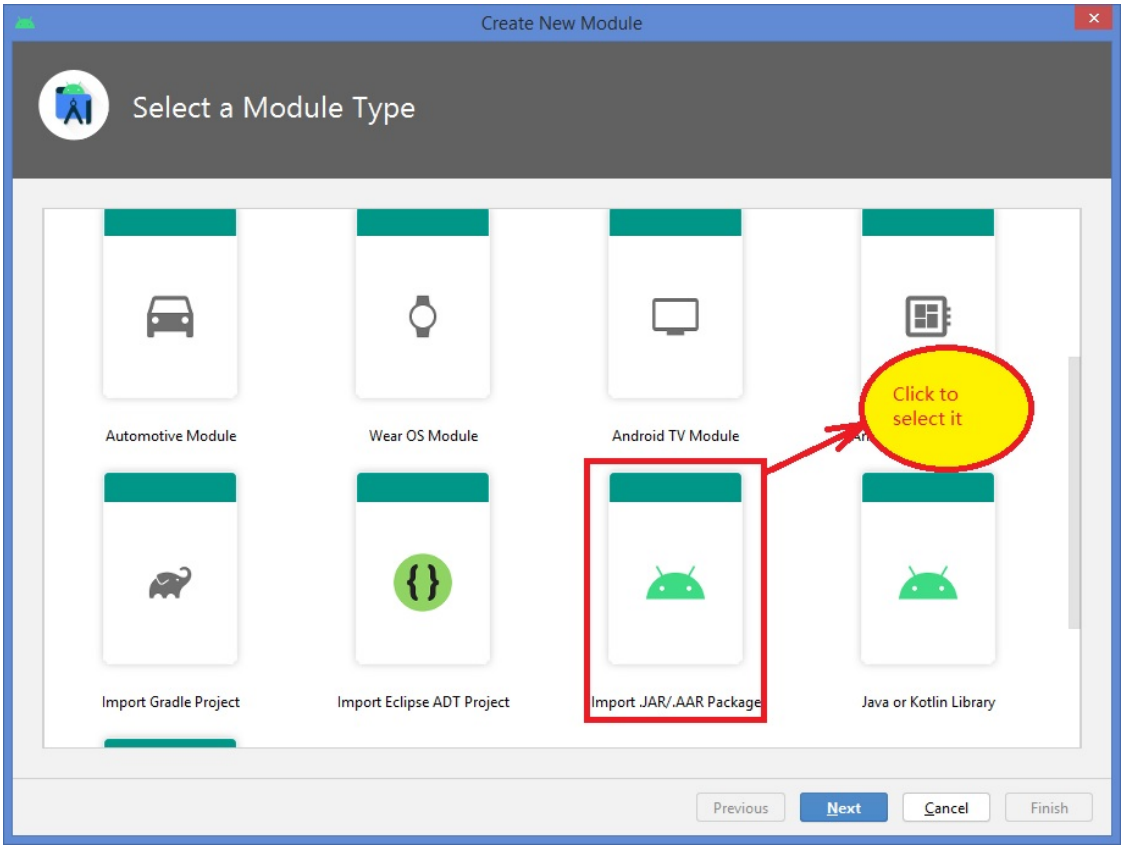
Install Library steps:

Step1: Download android aar library file : DFL168A.aar (http://dafulaielectronics.com/DFL168A.aar)

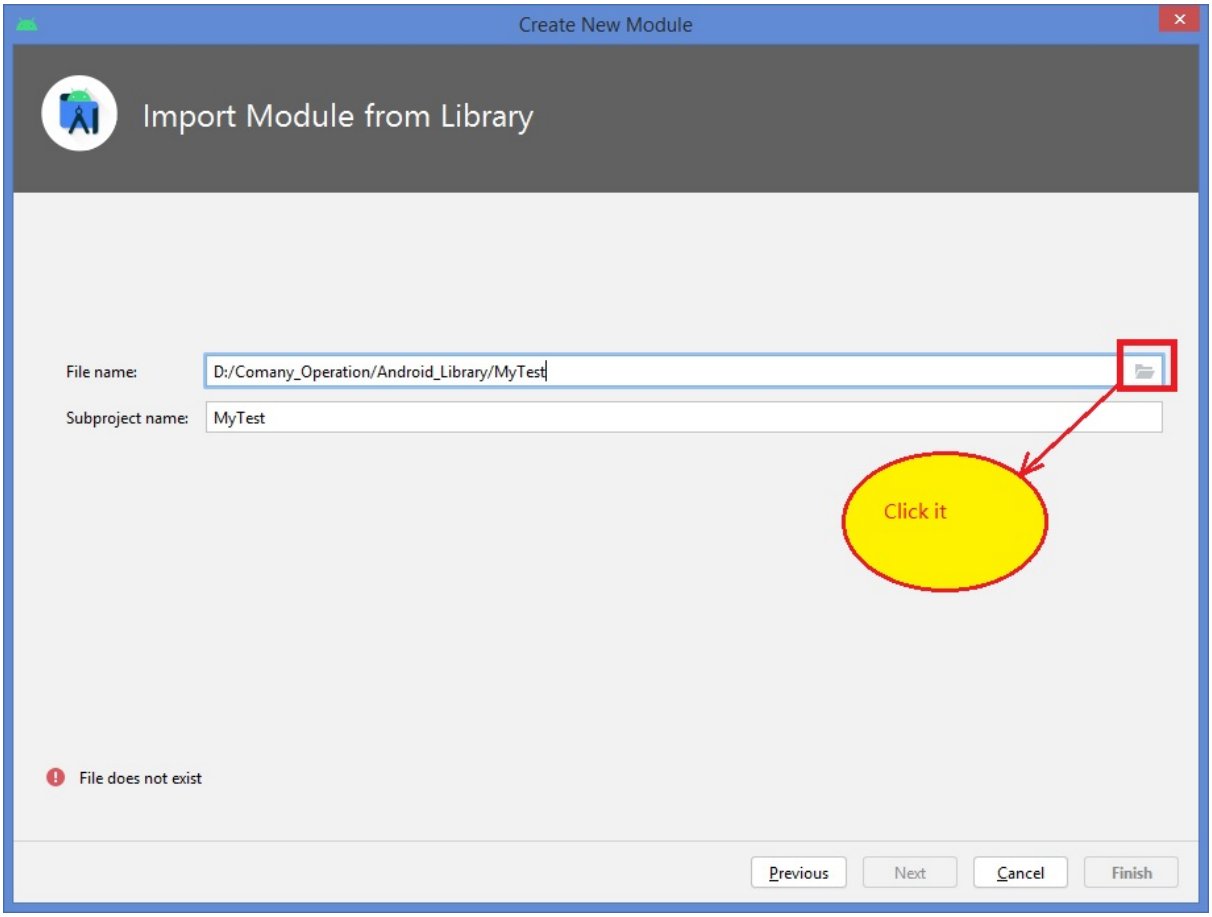
Step2: Run Android Studio 4.1.3 or higher, and open your Project. If you have no project, please set up new project. After you have project, click on menu "File / New Module ... " as below:



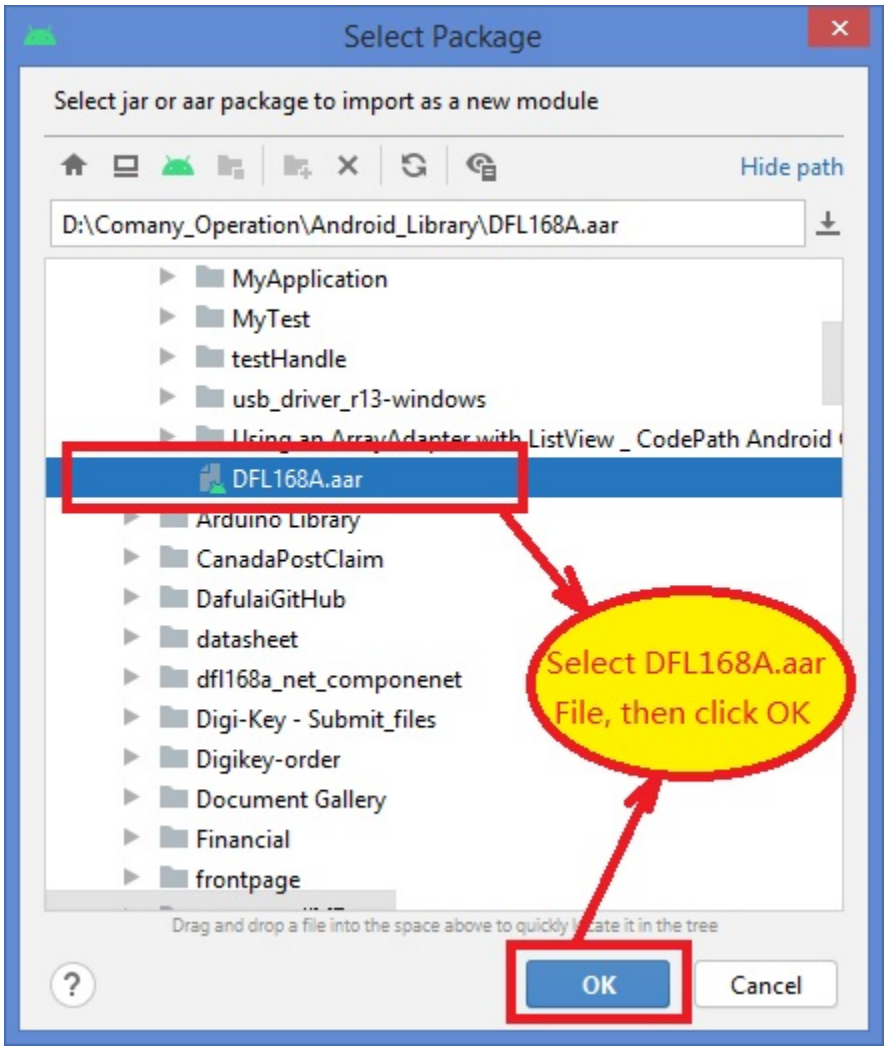
Step3: Click Icon "Import JAR/.AAR Package" to select it as below:



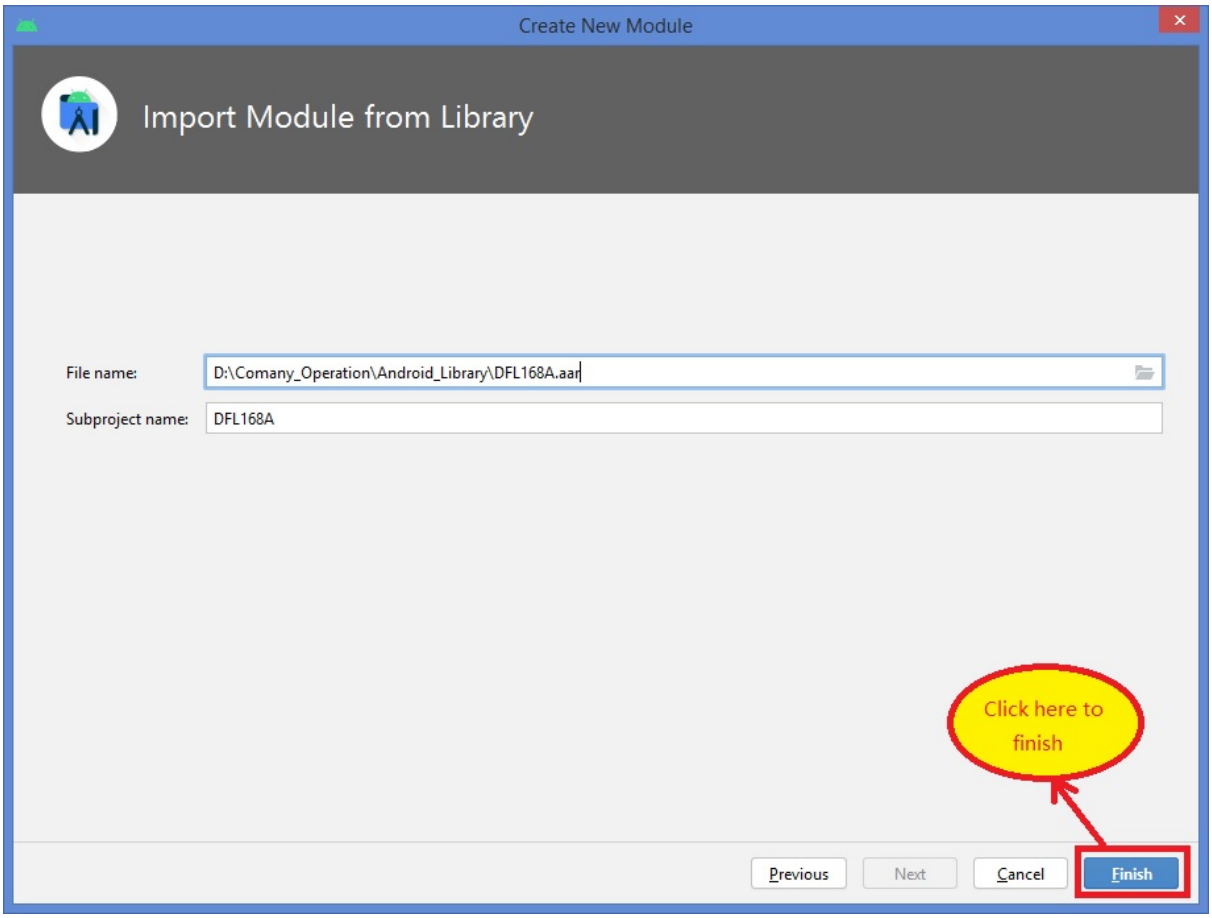
Step4: And then click Next button as above. The following window will occur, please click "browse file" icon as picture below:



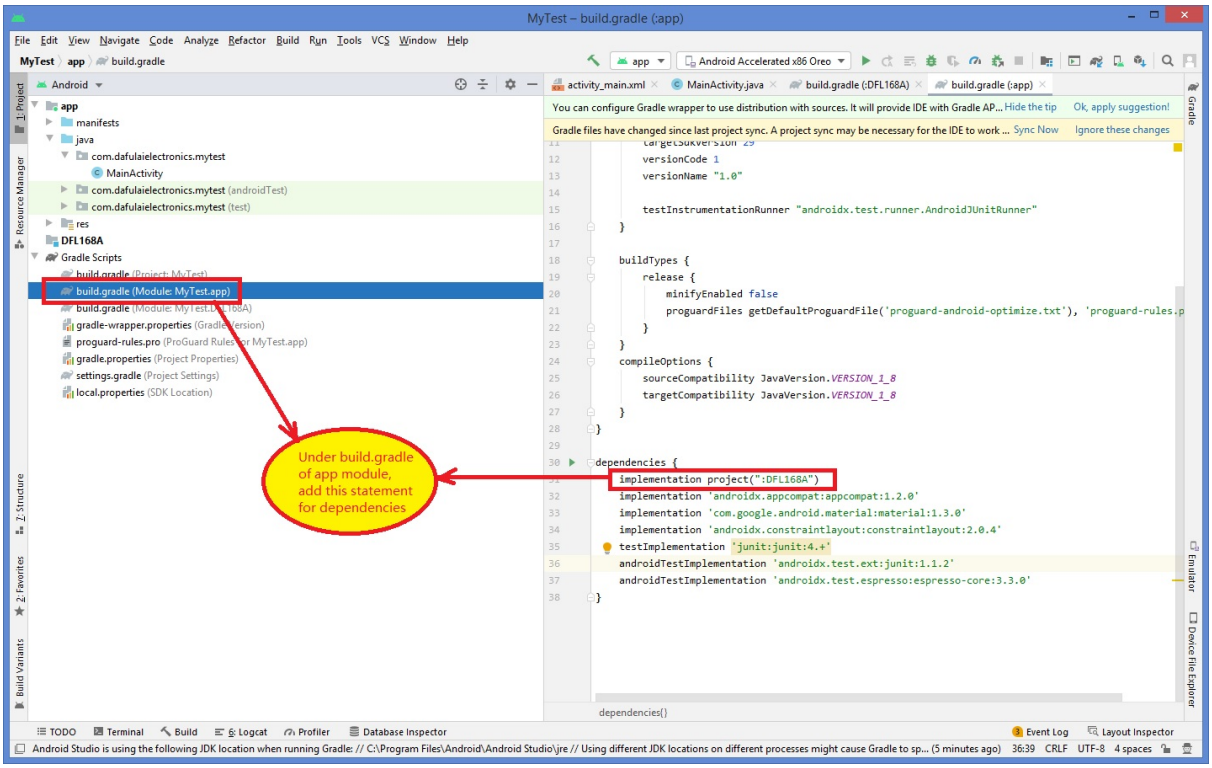
Now, you can see file browser window below:



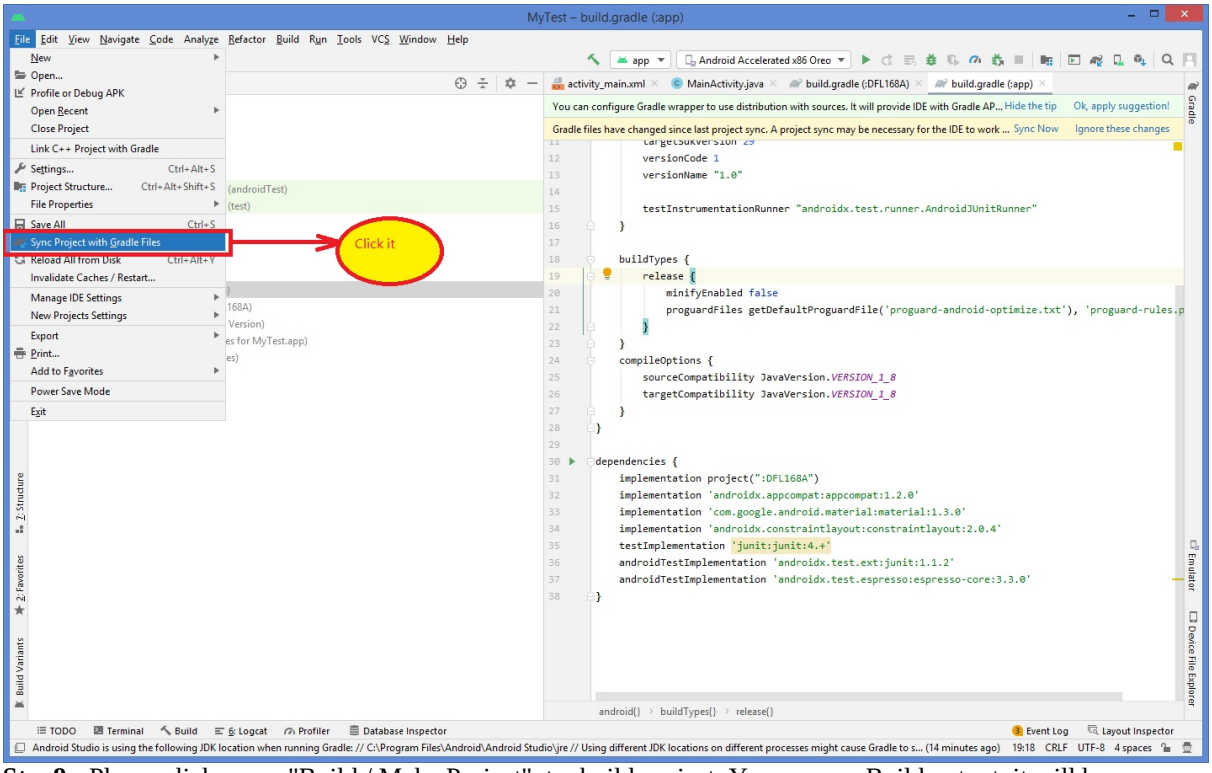
Step5: And then click OK button as above. The following window will occur, please click "Finish" button as picture below:



Step6: In build.gradle under Module app , in dependencies , add statement " implementation project(":DFL168A") " as screenshot below:



Step7: Please click "" as screenshot below:



Step8: Please click menu "Build / Make Project" to build project. You can see Build output, it will be successful. If you don't do this step, you may not see DFL168A method/member hint when you type source code.

4 How to use Library?

In general, you should create a DFL168A class instance (object), which contains a Handler member (handler), inside your activity.

After DFL168A instance does initialization to DFL168A IC successfully (by calling begin method of DFL168A instance) , you can create a Message class object instance , this message's "what" member value is set to special constant which denotes what vehicle parameter you want. For example, we want get VIN from ISO15765 protocol, message's "what" member value is 3048, it is not easy to remember this special value, you can use symbol constant DFL168A.ISO15765.GET_VIN to replace. And then you use handler of DFL168A to send this message (by calling sendMessage method of handler). When DFL168A instance get the parameter value from Vehicle successfully or fail in getting the parameter value, it will automatically call proper method of DFL168A_ResultProcess interface to give you the result. So your activity must implement DFL168A_ResultProcess interface in order to get what you want. Inside this interface method, you get result, and furthermore you can send another message by DFL168A handler to query another parameter,....

We give you detail steps below:

Step1. your activity which contains DFL168A object must implement DFL168A_ResultProcess interface.

Please add implements DFL168A_ResultProcess for your activity just like below:

```
public class MainActivity extends AppCompatActivity implements DFL168A_ResultProcess {
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
.....
}
```

The red text is added for supporting DFL168A_ResultProcess interface.

Step2. You should define member variable by your activity :

```
DFL168A mDFL168A=new DFL168A(this,DFL168A.J1939_PROTOCOL,1000,250000,960,1000);
```

For this construction above, you can read Construction Function

Now your code becomes the following content

```
public class MainActivity extends AppCompatActivity implements DFL168A_ResultProcess {
    DFL168A mDFL168A=new
```

```
DFL168A(this,DFL168A.J1939_PROTOCOL,1000,250000,960,1000);
@Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
  .....
}
```

J1939 Protocol is used in above code. If you want to use J1708, please change DFL168A.J1939_PROTOCOL to DFL168A.J1708_PROTOCOL.

If you want to use OBD2 with CAN BUS, please change DFL168A.J1939_PROTOCOL to DFL168A.*AUTO_PROTOCOL or just use* DFL168A mDFL168A=new DFL168A(this)

Now you have 2 different choices. One choice is that you use Bluetooth selection or scan graphic window by DFL168A class. If you choose this way, please go to step 3. The other choice is that you write your Bluetooth Select/Scan window code by yourself, in this way, please go to step 5

Step3: If you want to select Bluetooth without scan bluetooth device, please call the following method of DFL168A:

```
mDFL168A.selectBlutoothDevice();
```

For example, In your MainActivity - related layout, there is a button, its android:onClick ="selectBTNOScan".

You will add the following code in your class MainActivity

```
public void selectBTNOScan(View view) {
   mDFL168A.selectBlutoothDevice();
}
```

Your new code for class MainActivity becomes

```
public class MainActivity extends AppCompatActivity implements DFL168A_ResultProcess {
   DFL168A mDFL168A=new
   DFL168A(this,DFL168A.J1939_PROTOCOL,1000,250000,960,1000);
   @Override
   protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
   }
   public void selectBTNOScan(View view) {
       mDFL168A.selectBlutoothDevice();
   }
   .....
```

```
}
```

Similarly, you may want to select Bluetooth with scan bluetooth device, please call the following method of DFL168A:

```
mDFL168A.scanBlutoothDevice();
```

For example, In your MainActivity - related layout, there is another button, its android:onClick ="selectBTScan".

You will add the following code in your class MainActivity

```
public void selectBTScan(View view) {
   mDFL168A.scanBlutoothDevice();
}
```

Your new code for class MainActivity becomes

```
public class MainActivity extends AppCompatActivity implements DFL168A_ResultProcess {
DFL168A mDFL168A=new DFL168A(this,DFL168A.J1939_PROTOCOL,1000,250000,960,1000);
@Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
 public void selectBTNOScan(View view) {
  mDFL168A.selectBlutoothDevice();
}
public void selectBTScan(View view) {
  mDFL168A.scanBlutoothDevice();
}
.....
}
```

Step4: In your class MainActivity code editor window, press "Alt+Insert" key (or right click to pop up menu and click on "Generate ... ") to pop up menu, and click on "Override Method..." to pop up dialog, and select "onActivityResult" and click.

It will create code automatically below:

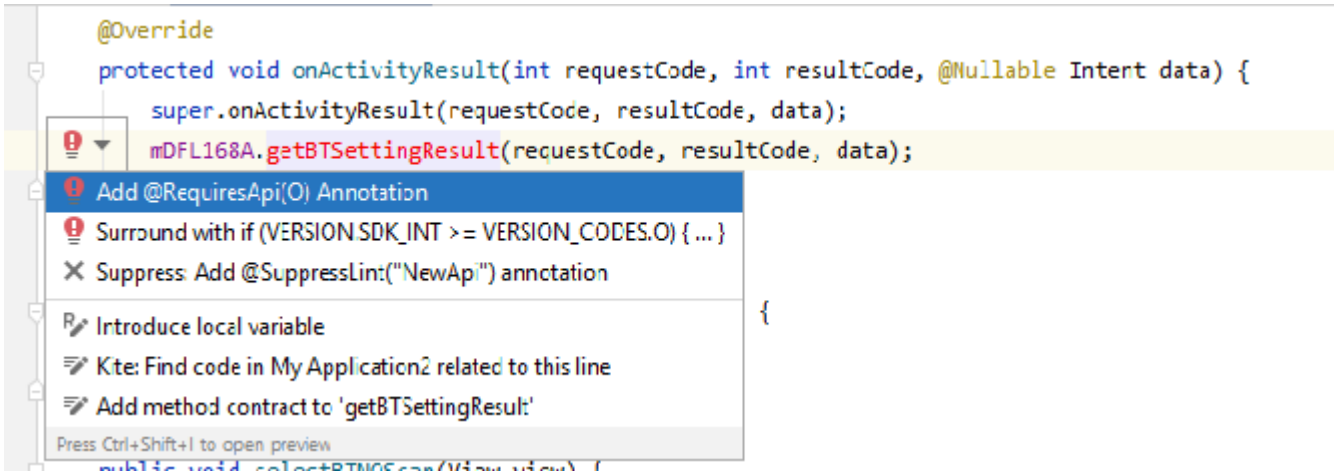
```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

}
```

The above method will be called automatically when user clicks bluetooth device in bluetooth devices list window.
Please add call DFL168A class getBTSettingResult method in above onActivityResult method. The above onActivityResult code becomes the following content:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    boolean Result;
    Result=mDFL168A.getBTSettingResult(requestCode, resultCode, data);
}
```

Of cause , you can use mDFL168A.getBTSettingResult(requestCode, resultCode, data, /*Intrude*/ false, /*Fast*/ true) to replace mDFL168A.getBTSettingResult(requestCode, resultCode, data).

You may get error for getBTSettingResult method call, please click on getBTSettingResult, and click on "Add@RequiresApi(O) Annotation" like screenshot below:



The code becomes

```
@RequiresApi(api = Build.VERSION_CODES.O)
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    boolean Result;
    Result=mDFL168A.getBTSettingResult(requestCode, resultCode, data);
}
```

We will tell you getBTSettingResult method detail later. One important thing is getBTSettingResult method called begin() method to initialize DFL168A IC. So you cannot call mDFL168A.begin() method again.

if boolean variable Result is true, DFL168A IC initialization is successful, now you can get vehicle data from vehicle by sending message (we will introduce in step 6).

If Result is false, you cannot access vehicle data, you can read mDFL168A.ErrorCode to know what reason for failure.

mDFL168A.ErrorCode value is Symbol constant int.

If you want to access Result in other methods, you can move boolean Result to MainActivity class member variable.

Your new code for class MainActivity becomes

```
public class MainActivity extends AppCompatActivity implements DFL168A_ResultProcess {
    boolean Result;
   DFL168A mDFL168A=new DFL168A(this,DFL168A.J1939_PROTOCOL,1000,250000,960,1000);
   @Override
      protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
      }
    public void selectBTNOScan(View view) {
      mDFL168A.selectBlutoothDevice();
    }
    public void selectBTScan(View view) {
      mDFL168A.scanBlutoothDevice();
    }
    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
    @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        Result=mDFL168A.getBTSettingResult(requestCode, resultCode, data);

    }
              .....
}
```

Now please go to step 6

Step5: change statement of DFL168A member variable (that is to say , first don't call construction function):

DFL168A mDFL168A;

After you get BluetoothDevice object by yourself (for example mBluetoothDevice), you will use "DFL168A mDFL168A=new DFL168A(this, mBluetoothDevice);" to set up mDFL168A object.

In above statement, you use Construction function below

```
DFL168A(Activity mActivity, BluetoothDevice mBluetoothDevice);
```

And default protocol is OBD2 with CAN BUS

Of cause , you can use the Construction function below

```
DFL168A(Activity mActivity, BluetoothDevice mBluetoothDevice,byte currProtocol,
```

```
int Timeout, int J1939Baudrate, int DEV_Baudrate, int DEV_timeout);
```

After you get DFL168A object: mDFL168A, You must call mDFL168A.begin() function to start DFL168A initialization , If success, it will return true. Otherwise return false. The returned value can be assigned to member variable Result in order to give other method use.

Result=mDFL168A.begin();

Or :

Result=mDFL168A.begin(intrude, fast);

Both intrude and fast are boolean argument, you can set true or false. The meaning will be explained in the DFL168A class method. Now go to step 6

Step6: In any button or menu 's click event handle function, you can add the following code to get DTC if you use J1939 protocol in DFL168A construction function.

```
if (Result)  {
    Message msg=new Message();
    msg.what=DFL168A.J1939.GET_DTC;
    msg.arg1=4;  //DTC format as Message.arg1: 1, 2, 3, 4
    mDFL168A.handler.sendMessage(msg);
}
```

And you should implement interface in your activity by following code in order to get DTC:

```
@Override
public void processDTC_J1939(boolean Cmd_Success, int DTC_Num, int[] SPN, int[] FMI, int[] CM, int[] OC) {
    if (Cmd_Success)
    {
        if (DTC_Num>=1) {
            Toast.makeText(this,"DTC(J1939) NUM: "+ DTC_Num+" SPN[0]: "+ SPN[0]+" FMI[0]: "+ FMI[0] ,Toast.LENGTH_SHORT).show();
        }
        if (DTC_Num>=2) {
            Toast.makeText(this," SPN[1]: "+ SPN[1]+" FMI[1]: "+ FMI[1] ,Toast.LENGTH_SHORT).show();
        }
        if (DTC_Num>=3) {
            Toast.makeText(this," SPN[2]: "+ SPN[2]+" FMI[2]: "+ FMI[2] ,Toast.LENGTH_SHORT).show();
        }
    }
    else {
```

```
        Toast.makeText(this,"DTC error(J1939)",Toast.LENGTH_SHORT).show();
    }
}
```

If you use J1708 protocol in DFL168A construction function, button or menu 's click event handle function becomes:

```
if (Result) {
    Message msg=new Message();
    msg.what=DFL168A.J1708.GET_DTC;
    mDFL168A.handler.sendMessage(msg);
}
```

And you should implement interface in your activity by following code in order to get DTC:

```
@Override
public void processDTC_J1708(boolean Cmd_Success, int DTC_Num, int MID, int[] PID_SID, boolean
[] IsPID, int[] FMI, boolean[] IsActive, boolean[] OccurrenceExist, int[] OccurrenceCount) {

    if (Cmd_Success)
    {

        Toast.makeText(this,"DTC Number: "+ DTC_Num +" MID:" + MID,Toast.LENGTH_SHORT
        ).show();
        if (DTC_Num>=1)
        {

            Toast.makeText(this,IsPID[0]?"PID[0]: "+PID_SID[0] :  "SID[0]:"+PID_SID[0],Toast.
            LENGTH_SHORT).show();

        }
        if (DTC_Num>=2)
        {

            Toast.makeText(this,IsPID[1]?"PID[1]: "+PID_SID[1] :  "SID[1]:"+PID_SID[1],Toast.
            LENGTH_SHORT).show();

        }

        }

    else {

        Toast.makeText(this,"DTC error(J1708)",Toast.LENGTH_SHORT).show();

    }

}
```

If you use ISO15765 or auto protocol in DFL168A construction function, button or menu 's click event

handle function becomes:

```
if (Result)  {
   Message msg=new Message();
   msg.what=DFL168A.ISO15765.GET_DTC;
   mDFL168A.handler.sendMessage(msg);
}
```

And you should implement interface in your activity by following code in order to get DTC:

```
@Override
public void processDTC(boolean Cmd_Success, int DTC_Num, String[] DTC) {
   if (Cmd_Success)
   {
      Toast.makeText(this,"DTC Number: "+ DTC_Num,Toast.LENGTH_SHORT).show();
      if (DTC_Num>=1)
      {
         Toast.makeText(this,"DTC[0]: "+ DTC[0],Toast.LENGTH_SHORT).show();
      }
      if (DTC_Num>=2)
      {
         Toast.makeText(this,"DTC[1]: "+ DTC[1],Toast.LENGTH_SHORT).show();
      }
      if (DTC_Num>=3)
      {
         Toast.makeText(this,"DTC[2]: "+ DTC[2],Toast.LENGTH_SHORT).show();
      }
      if (DTC_Num>=4)
      {
         Toast.makeText(this,"DTC[3]: "+ DTC[3],Toast.LENGTH_SHORT).show();
      }

   }
   else {
      Toast.makeText(this,"DTC error",Toast.LENGTH_SHORT).show();
   }

}
```

For vehicle's other parameter, our code is almost the same as "GET DTC" except different msg.what value

and different interface method.

For J1939 , it is special. All parameters except VIN DTC and clear DTC, we should send message PGN refresh (for example,PGN65269, you should set up msg.what=DFL168A.J1939.PGN65269.*REFRESH and then send message by "*mDFL168A.handler.sendMessage(msg);"), and in the related interface method to send parameter message.

For example, ambient temperature and barometric pressure parameters belong to PGN65269. In order to get ambient temperature and barometric pressure, we should add code below in button or menu 's click event handle function:

```
if (Result)  {
            Message msg=new Message();
            msg.what=DFL168A.J1939.PGN65269.REFRESH;
            mDFL168A.handler.sendMessage(msg);
}
```

And you should implement interface in your activity by following code in order to PGN fresh result:

```
@Override
public void processRefreshPGN65269_J1939(boolean Cmd_Success) {
    if (Cmd_Success)
    {
       Toast.makeText(this,"PGN 65269 refresh success" ,Toast.LENGTH_SHORT).show();
       Message msg=new Message();
        msg.what=DFL168A.J1939.PGN65269.GET_AMBIENT_TEMPERATURE;
       mDFL168A.handler.sendMessage(msg);
    }
    else {
       Toast.makeText(this,"PGN 65269 refresh error",Toast.LENGTH_SHORT).show();
    }
}
```

And you should implement interface in your activity by following code in order to get ambient temperature :

```
@Override
public void processAmbientTemperature_J1939(boolean Cmd_Success, float Temperature) {
   if (Cmd_Success)
   {
      Toast.makeText(this,"Ambient Temperature: "+Temperature ,Toast.LENGTH_SHORT).show();

   }
   else {
      Toast.makeText(this,"Ambient Temperature error",Toast.LENGTH_SHORT).show();
   }
   Message msg=new Message();
   msg.what=DFL168A.J1939.PGN65269.GET_BAROMETRIC_PRESSURE;
   mDFL168A.handler.sendMessage(msg);
```

```
}
```

And you should implement interface in your activity by following code in order to get barometric pressure :

```
@Override
public void processBarometricPressure_J1939(boolean Cmd_Success, float Pressure) {
    if (Cmd_Success)
    {
        Toast.makeText(this,"Barometric Pressure: "+Pressure ,Toast.LENGTH_SHORT).show();

    }
    else {
        Toast.makeText(this,"Barometric Pressure error",Toast.LENGTH_SHORT).show();
    }

}
```

Notes: *You must implement all method of DFL168A_ResultProcess interface even though you didn't use it. I n general , you can use short cut " Alt+Insert" to choose implement methods to create empty code automatically.*

5 DFL168A class

DFL168A Library has a class "DFL168A".

We use its member handler to send message, and we get result by DFL168A_ResultProcess interface method.

We use the following member and methods:

1 Construction function to set up an instance.

2 begin method to initialize DFL168A IC

3 end method to release DFL168A IC resource

4 Handler type of member: handler which is used to send message.

5 member Uart , which is equivalent serial port of bluetooth. We used it for access DFL168A UART2 in transparent mode.

6. int member ErrorCode, which tells us Error reason when we fail in initializing DFL168A IC by calling begin method

We will explain these members and methods above.

5.1 Members

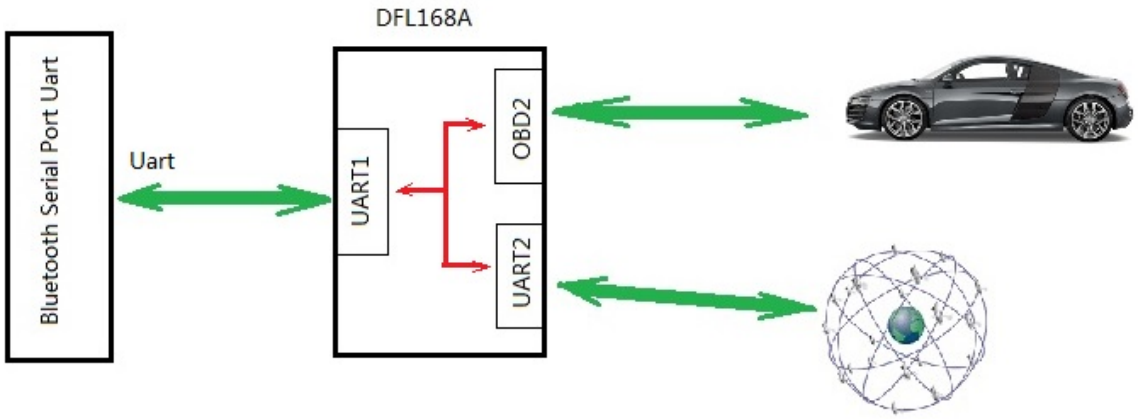
1. public Handler handler;

This is instance of Handler. We use handler.sendMessage(msg) method to send request to DFL168A IC. The msg variable is Message class. msg.what denotes what parameter of vehicle you want (Please read Constant) . Sometimes, DFL168A needs more information for this request, you can provide from msg.arg1 and/or msg.arg2. When DFL168A IC gets parameter from vehicle or fail in getting result, it will call method of DFL168A_resultProcess interface automatically. And you can get result from this method from interface.

Some parameters are not from vehicle such as I-button ID, GPS, and Analog/Discrete Input, and Discrete output, we still use Message/Handler to get DFL168A information.

2. public uart Uart;

This is useful when we use Transparent mode to access DFL168A DEV1 (Uart2 of DFL168A IC). Actually uart class is Bluetooth serial port. Uart is an instance of uart. Uart connects UART1 of dfl168A. Please see picture below for serial port relation:



You don't need to use Uart instance because we provide Message/Handler way to get all information. However, when UART2 of DFL168A IC connects a general serial device (Usually, UART2 of DFL168A IC connects GPS module.) , and if you use transparent mode which makes UART1 of dfl168A connect UART2 of dfl168A directly, you can use Uart (Bluetooth) access your general serial device which connects UART2 of DFL168A IC.

We introduce some public members and methods of uart class below:

```
public boolean success;
success is true -------- Bluetooth serial port is OK.
success is false -------- Bluetooth serial port is not available.

public int available();
Get the number of bytes (characters) available for reading from the serial port.

public void clearRxd();
Empty serial port all receiver data.

public int peek();
Returns the next byte (character) of incoming serial data without removing it
from the internal serial buffer, -1 denotes no data available.

public int read();
return the first byte of incoming serial data available (or -1 if no data is
available)

public void setTimeout(long timeout);
sets the maximum milliseconds to wait for serial data. It defaults to 1000
milliseconds.
```

```
public int readBytes(byte[] myBuffer, int length);
reads characters from the serial port into a myBuffer. The function terminates if
the determined length has been read, or it times out.
Returned value is actually how many bytes been read.

public int readBytesUntil(char character, byte[] myBuffer, int length);
reads characters from the serial port into an array myBuffer. The function
terminates (checks being done in this order) if the determined length has been
read, if it times out , or if the terminator character is detected (in which case
array myBuffer gets the characters up to the last character before the supplied
terminator). The terminator itself is not returned in the myBuffer. The
terminator character is discarded from the serial port.

public String readString();
reads characters from the serial port into a String, and returns this String. The
function terminates if it times out

public String readStringUntil(char character);
reads characters from the serial port into a String, and returns this String. The
function terminates if it times out or if the terminator character is detected.
The terminator character is discarded from the serial port.

public String readExistingString();
reads all characters from the serial port into a String, and returns this String.
This function will return String immediately if no any data available from the
serial port.

public void print(String str);
Write String str into the serial port's transmit buffer.

public void print(String str);
Write String str and "\r\n" into the serial port's transmit buffer.

public void write(byte val);
Write byte val into the serial port's transmit buffer.

public void write(byte[] buf, int len);
Write byte array buf (using the length len) into the serial port's transmit
buffer.

public void flush();
Waits for the transmission of outgoing serial data to complete.
```

3. public int ErrorCode

When we call "begin" method of DFL168A instance to initialize DFL168A IC, this method will return true when successes. However, when returned false, we know IC fail in initialization. If we want to know the failure reason, we must read variable ErrorCode value. We use symbol constant denotes the meaning:

ERROR_NO_UART ----------value is 0, denotes that bluetooth cannot become valid uart

ERROR_THREAD_SLEEP----------value is 1, denotes that sleep delay exception occurs.
ERROR_INVALID_DFL168A_IC---------value is 2, denotes that invalid DFL168A IC is used.
ERROR_AT_COMMAND----------value is 3, denotes that AT Command cannot run correctly
ERROR_WRONG_PROTOCOL----------value is 4, denotes that vehicle does not support current selected protocol.

5.2 Constant

For protocol name, we have the following constant in the DFL168A library:

AUTO_PROTOCOL---------- This is ISO15765 protocol, and DFL168A will select correct specific ISO15765 protocol automatically, such as 11bits or 29 bits and baud rate.
ISO15765_11_500_PROTOCOL----------This is ISO15765 protocol with 11 bits CAN ID and 500Kbps baud rate.
ISO15765_29_500_PROTOCOL----------This is ISO15765 protocol with 29 bits CAN ID and 500Kbps baud rate.
ISO15765_11_250_PROTOCOL----------This is ISO15765 protocol with 11 bits CAN ID and 250Kbps baud rate.
ISO15765_29_250_PROTOCOL----------This is ISO15765 protocol with 29 bits CAN ID and 250Kbps baud rate.
J1939_PROTOCOL----------This is J1939 protocol, and baud rate will be decided by your construction function.
J1708_PROTOCOL----------This is J1708/J1587 protocol,

You access above constant by "DFL168A. prefix" such as "DFL168A.AUTO_PROTOCOL"

For member ErrorCode, its values are the constant below:
ERROR_NO_UART ----------value is 0, denotes that bluetooth cannot become valid uart
ERROR_THREAD_SLEEP----------value is 1, denotes that sleep delay exception occurs.
ERROR_INVALID_DFL168A_IC---------value is 2, denotes that invalid DFL168A IC is used.
ERROR_AT_COMMAND----------value is 3, denotes that AT Command cannot run correctly
ERROR_WRONG_PROTOCOL----------value is 4, denotes that vehicle does not support current selected protocol.

You access above constant by "DFL168A. prefix" such as "DFL168A.ERROR_NO_UART"

For Message.what value, we can put it into different type of constant symbol.

1 For general IC access such as GPIO analog input, one wire ID, transparent mode, we have the following constants:

GET_ONE_WIRE_ID ----------denotes you want to get one wire iButton ID, its interface method is void processOnewireID(boolean Cmd_Success, int[] ID);

GET_DIN ----------denotes you want to get digital Input. Which digital input is decided by message.arg1. For example, DIN1, message.arg1=1. The relative interface method is `void processDIN(boolean Cmd_Success,int port, boolean Value);`

SET_DOUT----------denotes you want to set digital output, digital output number is set by message.arg1, value is set by message.arg2, 1 is logic high, 0 is logic low. The relative interface method is void processDOUT(int port, boolean Value);

GET_ANALOG----------denotes you want to get analog input. The relative interface method is "`void processAnalogInput(boolean Cmd_Success, float AnalogInputValue);`"

BEGIN_TRANSPARENT_UART------denotes you want to begin transparent mode. The relative interface method is "`void processBeginTransparentSerial();`"

END_TRANSPARENT_UART---------denotes you want to end transparent mode. The relative interface method is "`void processEndTransparentSerial(boolean Cmd_Success);`"

SET__EXIT_TRANSPARENT_KEY--denotes you want to set key code for exiting transparent mode , key code is set by message.arg1 (0 to 255). The relative interface method is "`void processSetExitTransparentKey( boolean Cmd_Success);`"

SET_SLEEP_DELAY----------denotes you want to set IC enters sleep delay time , delay time is set by message.arg1 (0 to 65535 Seconds). The relative interface method is "`void processSetExitTransparentKey(boolean Cmd_Success);`"

The above constants are set by "DFL168A. prefix", for example, "Message msg=new Message(); msg. what=DFL168A.GET_ONE_WIRE_ID; "

2 For ISO15765, we have the following constants:

ISO15765.GET_COOLANT_TEMPERATURE----------denotes you want to get coolant temperature in Celsius. The relative interface method is "`void processCoolantTemperature(boolean Cmd_Success,float temperature);`"

ISO15765.GET_ENGINE_SPEED----------denotes you want to get engine speed in rpm. The relative interface method is "`void processEngineSpeed(boolean Cmd_Success, float EngineSpeed);`"

ISO15765.GET_VEHICLE_SPEED----------denotes you want to get vehicle speed in km/h. The relative interface method is "`void processVehicleSpeed(boolean Cmd_Success,float VehicleSpeed);`"

ISO15765.GET_INTAKE_MANIFOLD_PRESSURE-------denotes you want to get manifold pressure derived from a Manifold Absolute Pressure sensor in kPa. The relative interface method is "`void processIntakeManifoldPressure( boolean Cmd_Success,float IntakeManifoldPressure);`"

ISO15765.GET_FUEL_SYSTEM_STATUS----------denotes you want to get fuel system status. The relative interface method is "`processFuelSystemStatus(boolean Cmd_Success,boolean A_Openloop, boolean A_Closedloop, boolean A_OpenloopByDriving_Con,boolean A_OpenloopByFault,boolean A_ClosedloopButFault, boolean B_Openloop, boolean B_Closedloop,boolean B_OpenloopByDriving_Con,boolean B_OpenloopByFault,boolean B_ClosedloopButFault); `"

ISO15765.GET_CALCULATED_LOAD_VALUE----------denotes you want to get calculated LOAD Value in percentage. The relative interface method is "`void processCalculatedLoadValue(boolean Cmd_Success,float CalculatedLoad);`"

ISO15765.GET_SHORT_TERM_FUEL_TRIM_BANK13-------denotes you want to get short term fuel trim in percentage. The relative interface method is "`void processShortTermFuelTrimBank13(boolean Cmd_Success,float Bank1,float Bank3);`"

ISO15765.GET_LONG_TERM_FUEL_TRIM_BANK13---------denotes you want to get long term fuel trim in percentage. The relative interface method is "`void processLongTermFuelTrimBank13(boolean Cmd_Success,float Bank1,float Bank3);`"

ISO15765.GET_SHORT_TERM_FUEL_TRIM_BANK24-------denotes you want to get short term fuel trim in percentage. The relative interface method is "`void processShortTermFuelTrimBank24(boolean Cmd_Success,float Bank2,float Bank4);`"

ISO15765.GET_LONG_TERM_FUEL_TRIM_BANK24---------denotes you want to get long term fuel trim in percentage. The relative interface method is "`void processLongTermFuelTrimBank24(boolean Cmd_Success,float`

Bank2,float Bank4);"

ISO15765.GET_IGNITION_TIMING_ADVANCE----------denotes you want to get Ignition timing spark advance in degrees before top dead center (°BTDC) for #1 cylinder (not including mechanical advance). The relative interface method is "void processIgnitionTimingAdvance(boolean Cmd_Success,float Angle);"

ISO15765.GET_INTAKE_AIR_TEMPERATURE----------denotes you want to get intake manifold air temperature in Celsius degree. The relative interface method is "void processIntakeAirTemperature(boolean Cmd_Success,float temperature);"

ISO15765.GET_AIR_FLOW_RATE_FRM_MAF----------denotes you want to get air flow rate from mass air flow sensor in g/s. The relative interface method is "void processAirFlowRateFrmMAF(boolean Cmd_Success,float FlowRate);"

ISO15765.GET_ABS_THROTTLE_POS----------denotes you want to get absolute throttle position in percentage. The relative interface method is "void processAbsThrottlePosition(boolean Cmd_Success,float Percent);"

ISO15765.GET_OXYGEN_SENSOR_LOCATION----------denotes you want to get location of oxygen sensors. The relative interface method is "void processOxygenSensorLocation(boolean Cmd_Success,boolean Bank1_Sensor1Present, boolean Bank1_Sensor2Present, boolean Bank1_Sensor3Present,boolean Bank1_Sensor4Present, boolean Bank3_Sensor1Present, boolean Bank3_Sensor2Present, boolean Bank2_Sensor1Present, boolean Bank2_Sensor2Present, boolean Bank2_Sensor3Present,boolean Bank2_Sensor4Present, boolean Bank4_Sensor1Present, boolean Bank4_Sensor2Present);"

ISO15765.GET_BANK10_SENSOR1_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 1 – Sensor 1. The relative interface method is "void processBank1OSensor1Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK10_SENSOR2_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 1 – Sensor 2. The relative interface method is "void

processBank1OSensor2Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK10_SENSOR3_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 1 – Sensor 3. The relative interface method is "void processBank1OSensor3Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK10_SENSOR4_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 1 – Sensor 4. The relative interface method is "void processBank1OSensor4Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK20_SENSOR1_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 2 – Sensor 1. The relative interface method is "void processBank2OSensor1Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK20_SENSOR2_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 2 – Sensor 2. The relative interface method is "void processBank2OSensor2Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK20_SENSOR3_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 2 – Sensor 3. The relative interface method is "void processBank2OSensor3Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK20_SENSOR4_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 2 – Sensor 4. The relative interface method is "void processBank2OSensor4Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK30_SENSOR1_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 3 – Sensor 1. The relative interface method is "void processBank3OSensor1Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK30_SENSOR2_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 3 – Sensor 2. The relative interface method is "void processBank3OSensor2Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_BANK40_SENSOR1_VOLTAGE----------denotes you want to get 0 to 1 volt oxygen sensor for Bank 4 – Sensor 1. The relative interface method is "void processBank4OSensor1Voltage(boolean Cmd_Success,float

OutVoltage);"

ISO15765.GET_BANK40_SENSOR2_VOLTAGE---------denotes you want to get 0 to 1 volt oxygen sensor for Bank 4 – Sensor 2. The relative interface method is "void processBank4OSensor2Voltage(boolean Cmd_Success,float OutVoltage);"

ISO15765.GET_OBD_CERTIFIED----------denotes you want to get OBD requirements String to which vehicle or engine is certified. The relative interface method is "void processOBDCertified(boolean Cmd_Success, String OBD); "

ISO15765.GET_TIME_SINCE_ENGINE_START---------denotes you want to get time Since Engine Start in seconds. For non-hybrid vehicles, TotalTime shall increment while the engine is running. It shall freeze if the engine stalls. TotalTime shall be reset to zero during every control module power-up and when entering the key-on, engine off position. TotalTime is limited to 65535 seconds and shall not wrap around to zero. The relative interface method is "void processTimeSinceEngineStart(boolean Cmd_Success,float TotalTime);"

ISO15765.GET_DISTANCE_TRAVELED_MIL----------denotes you want to get the distance (Km) traveled while MIL is activated. The relative interface method is "void processDistanceTraveledMIL(boolean Cmd_Success,float Distance);"

ISO15765.GET_FUEL_RAIL_PRESSURE----------denotes you want to get the fuel rail pressure (kPa) at the engine when the reading is referenced to atmosphere (gage pressure). The relative interface method is "void processFuelRailPressure(boolean Cmd_Success,float Pressure); "

ISO15765.GET_FUEL_LEVEL_INPUT----------denotes you want to get the nominal fuel tank liquid fill capacity as a percent of maximum. The relative interface method is "void processFuelLevelInput(boolean Cmd_Success, float Percent);"

ISO15765.GET_DISTANCE_TRAVELED_SINCE_DTC_CLEAR-------denotes you want to get the distance (Km) accumulated since DTCs were cleared (via external test equipment or possibly, a battery disconnect). The relative interface method is "void processDistanceTraveledSinceDTC_Clear(boolean Cmd_Success,float Distance);"

ISO15765.GET_BAROMETRIC_PRESSURE----------denotes you want to get the barometric pressure in kPa. The relative interface method is "void

processBarometricPressure(boolean Cmd_Success,float Pressure);"

ISO15765.GET_CONTROL_MODULE_VOLTAGE------denotes you want to get the control module voltage. The relative interface method is "void processControlModuleVoltage(boolean Cmd_Success,float Voltage);"

ISO15765.GET_RELATIVE_THROTTLE_POS----------denotes you want to get the relative throttle position in percentage. The relative interface method is "void processRelativeThrottlePos(boolean Cmd_Success,float Percent);"

ISO15765.GET_AMBIENT_TEMPERATURE----------denotes you want to get the ambient air temperature in Celsius degree. The relative interface method is "void processAmbientTemp(boolean Cmd_Success,float AmbientTemp);"

ISO15765.GET_COMMANDED_THROTTLE_ACTUATOR_CONTROL------denotes you want to get the commanded throttle actuator control in percentage. The relative interface method is "void processCommandedThrottleActuatorControl(boolean Cmd_Success,float Percent);"

ISO15765.GET_ENGINE_RUNTIME_MIL----------denotes you want to get the engine run time (minutes) while MIL is activated. The relative interface method is "void processEngineRunTimeMIL(boolean Cmd_Success,float TotalTime); "

ISO15765.GET_ENGINE_RUNTIME_SINCE_DTC_CLEAR---------denotes you want to get the engine run time (minutes) since DTCs cleared. The relative interface method is "void processEngineRunTimeSinceDTC_Clear(boolean Cmd_Success,float TotalTime);"

ISO15765.GET_TYPE_OF_FUEL_USED_CURRENTLY----------denotes you want to get the type of fuel currently being utilized by the vehicle. The relative interface method is "void processTypeOfFuelUsedCurrently(boolean Cmd_Success,String FuelType); "

ISO15765.GET_RELATIVE_ACCELERATOR_PEDAL_POS------denotes you want to get the relative accelerator pedal position in percentage. The relative interface method is "void processRelativeAcceleratorPedalPosition(boolean Cmd_Success,float Percent); "

ISO15765.GET_HYBRID_BATTERY_PACK_REMAINING_LIFE----denotes you want to get the percent remaining life for the hybrid battery pack. The relative interface

method is "`void processHybridBatteryPackRemainingLife( boolean Cmd_Success,float Percent);`"

ISO15765.GET_ENGINE_OIL_TEMPERATURE---------denotes you want to get the engine oil temperature in Celsius degree. The relative interface method is "`void processEngineOilTemperature(boolean Cmd_Success,float Temperature);`"

ISO15765.GET_FUEL_RATE---------denotes you want to get the amount of fuel consumed by engine per unit of time in liters per hour. The relative interface method is "`void processFuelRate(boolean Cmd_Success, float FuelRate);`"

ISO15765.GET_ACTUAL_ENGINE_TORQUE----------denotes you want to get the actual engine - percent torque. The relative interface method is "`void processActualEngineTorque(boolean Cmd_Success,float ActualEngineTorque);`"

ISO15765.GET_MIL_STATUS---------denotes you want to get the Malfunction Indicator Lamp (MIL) status. The relative interface method is "`void processMILStatus(boolean Cmd_Success,boolean MIL_IS_ON);`"

ISO15765.GET_ENGINE_RUN_TIME----------denotes you want to get the engine run time in secconds. The relative interface method is "`void processEngineRunTime(boolean Cmd_Success,float TotalEngineRunTime, float TotalIdleRunTime,float TotalRunTimeWithPTOActive);`"

ISO15765.GET_VIN----------denotes you want to get the Vehicle Identification Number (VIN) as assigned by the vehicle manufacturer. The relative interface method is "`void processVIN(boolean Cmd_Success, String VIN);`"

ISO15765.GET_DTC----------denotes you want to get get all DTCs. The relative interface method is "`void processDTC(boolean Cmd_Success,int DTC_Num,String DTC[]);`"

ISO15765.CLEAR_DTC----------denotes you want to clear all DTCs. The relative interface method is "`void processClearDTC(boolean Cmd_Success);`"

The above constants are set by "DFL168A. prefix", for example, "Message msg=new Message(); msg. what=DFL168A.ISO15765.GET_COOLANT_TEMPERATURE; "

3 For J1939, we have the following constants:

J1939..GET_VIN----------denotes you want to get 19 characters' VIN number from vehicle. The relative interface method is "`void processVIN_J1939(boolean Cmd_Success,String VIN);`"

J1939..GET_DTC----------denotes you want to get DTC information of vehicle. DTC format is set by message.arg1 which must be 1, or 2, or 3, or 4. The relative interface method is "`void processDTC_J1939(boolean Cmd_Success,int DTC_Num,int SPN[], int FMI[],int CM[], int OC[]);`"

J1939..CLEAR_DTC----------denotes you want to clear DTC of vehicle. The relative interface method is "`void processClearDTC_J1939(boolean Cmd_Success);`"

J1939.PGN65267.REFRESH-------denotes you want to refresh PGN65267 data from vehicle. The relative interface method is "`void processRefreshPGN65267_J1939(boolean Cmd_Success);`"

J1939.PGN65267.GET_LATITUDE----denotes you want to get Latitude from vehicle. The relative interface method is "`void processLatitude_J1939(boolean Cmd_Success,float Latitude);`"

J1939.PGN65267.GET_LONGITUDE----denotes you want to get Longitude from vehicle. The relative interface method is "`void processLongitude_J1939(boolean Cmd_Success,float Longitude);`"

J1939.PGN65262.REFRESH----------denotes you want to refresh PGN65262 data from vehicle. The relative interface method is "`void processRefreshPGN65262_J1939(boolean Cmd_Success);`"

J1939.PGN65262.GET_COOLANT_TEMPERATUR------denotes you want to get engine coolant temperature in Celsius degree. The relative interface method is "`void processCoolantTemperature_J1939(boolean Cmd_Success,float Temperature);`"

J1939.PGN65262.GET_FUEL_TEMPERATURE----------denotes you want to get fuel temperature in Celsius degree. The relative interface method is "`void processFuelTemperature_J1939(boolean Cmd_Success,float Temperature);`"

J1939.PGN65262.GET_OIL_TEMPERATURE----------denotes you want to get engine oil temperature in Celsius degree. The relative interface method is "`void processOilTemperature_J1939(boolean Cmd_Success,float Temperature);`"

J1939.PGN65256.REFRESH----------denotes you want to get refresh PGN65256 data from vehicle. The relative interface method is "`void processRefreshPGN65256_J1939( boolean Cmd_Success);`"

J1939.PGN65256.GET_ALTITUDE----------denotes you want to get Altitude from vehicle. The relative interface method is "void processAltitude_J1939(boolean Cmd_Success,float Altitude);"

J1939.PGN65256.GET_NAV_BASED_SPEED---------denotes you want to get vehicle speed based on navigation in Km/h. The relative interface method is "void processNavBasedSpeed_J1939(boolean Cmd_Success,float Speed);"

J1939.PGN65269.REFRESH----------denotes you want to refresh PGN65269 data from vehicle. The relative interface method is "void processRefreshPGN65269_J1939(boolean Cmd_Success);"

J1939.PGN65269.GET_BAROMETRIC_PRESSURE----------denotes you want to get Barometric Pressure in KPa. The relative interface method is "void processBarometricPressure_J1939(boolean Cmd_Success,float Pressure);"

J1939.PGN65269.GET_AMBIENT_TEMPERATURE----------denotes you want to get Ambient Air Temperature in Celsius degree. The relative interface method is "void processAmbientTemperature_J1939(boolean Cmd_Success,float Temperature);"

J1939.PGN65269.GET_INLET_TEMPERATURE----------denotes you want to get Engine Air Inlet Temperature in Celsius degree. The relative interface method is "void processInletTemperature_J1939(boolean Cmd_Success, float Temperature);"

J1939.PGN65269.GET_ROAD_TEMPERATURE----------denotes you want to get road Temperature in Celsius degree. The relative interface method is "void processtRoadTemperature_J1939(boolean Cmd_Success,float Temperature);"

J1939.PGN65269.GET_CAB_INTERIOR_TEMPERATURE--------denotes you want to get Cab Interior Temperature in Celsius degree. The relative interface method is "void processCabInteriorTemperature_J1939(boolean Cmd_Success,float Temperature);"

J1939.PGN65257.REFRESH----------denotes you want to refresh PGN65257 data from vehicle. The relative interface method is "void processRefreshPGN65257_J1939(boolean Cmd_Success);"

J1939.PGN65257.GET_ENGINE_TRIP_FUEL----------denotes you want to get Engine Trip Fuel in L. The relative interface method is "void processEngineTripFuel_J1939(boolean Cmd_Success,float EngineTripFuel);"

J1939.PGN65257.GET_ENGINE_TOTAL_FUEL_USED----------denotes you want to get Engine Total Fuel Used in L. The relative interface method is "void

processEngineTotalFuelUsed_J1939(boolean Cmd_Success,float EngineTotalFuelUsed);"

J1939.PGN61444.REFRESH----------denotes you want to refresh PGN61444 data from vehicle. The relative interface method is "void processRefreshPGN61444_J1939(boolean Cmd_Success);"

J1939.PGN61444.GET_ACTUAL_ENGINE_TORQUE----------denotes you want to get Actual Engine - Percent Torque. The relative interface method is "void processActualEngineTorque_J1939(boolean Cmd_Success,float ActualEngineTorque);"

J1939.PGN61444.GET_ENGINE_SPEED----------denotes you want to get engine speed in rpm. The relative interface method is "void processEngineSpeed_J1939(boolean Cmd_Success,float EngineSpeed);"

J1939.PGN61443.REFRESH----------denotes you want to refresh PGN61443 data from vehicle. The relative interface method is "void processRefreshPGN61443_J1939(boolean Cmd_Success);"

J1939.PGN61443.GET_ACCEL_PEDAL_POSI1----------denotes you want to get Accelerator Pedal Position 1 in percentage. The relative interface method is "void processAccelPedalPosi1_J1939(boolean Cmd_Success,float AccelPedalPosi1);"

J1939.PGN61443.GET_ACCEL_PEDAL_POSI2----------denotes you want to get Accelerator Pedal Position 2 in percentage. The relative interface method is "void processAccelPedalPosi2_J1939(boolean Cmd_Success,float AccelPedalPosi2);"

J1939.PGN61443.GET_ENGINE_PERLOAD_AT_CURRENT_SPEED---------denotes you want to get Engine Percent Load At Current Speed. The relative interface method is "void processEnginePerLoadAtCurrSpeed_J1939(boolean Cmd_Success,float EnginePerLoadAtCurrSpeed);"

J1939.PGN65270.REFRESH----------denotes you want to refresh PGN65270 data from vehicle. The relative interface method is "void processRefreshPGN65270_J1939(boolean Cmd_Success);"

J1939.PGN65270.GET_INTAKE_MANIFOLD_PRESSURE----------denotes you want to get Engine Intake Manifold #1 Pressure in kPa. The relative interface method is "void processIntakeManifoldPressure_J1939(boolean Cmd_Success,float Pressure);"

J1939.PGN65270.GET_INTAKE_MANIFOLD_TEMPERATURE----------denotes you want to get Engine Intake Manifold 1 Temperature in Celsius degree. The relative interface method is "void processIntakeManifoldTemperature_J1939(boolean

Cmd_Success,float Temperature);"

J1939.PGN65270.GET_ENGINE_AIR_INLET_PRESSURE----------denotes you want to get Engine Air Inlet Pressure in kPa. The relative interface method is "void processEngineAirInletPressure_J1939(boolean Cmd_Success,float EngineAirInletPressure);"

J1939.PGN65270.GET_ENGINE_EXHAUST_GAS_TEMPERATURE----------denotes you want to get Engine Exhaust Gas Temperature in Celsius degree. The relative interface method is "void processEngineExhaustGasTemperature_J1939(boolean Cmd_Success,float Temperature);"

J1939.PGN65270.GET_ENGINE_AIR_FILTER_DIFF_PRESSURE----------denotes you want to get Engine Air Filter 1 Differential Pressure in kPa. The relative interface method is "void processEngineAirFilterDiffPressure_J1939(boolean Cmd_Success,float EngineAirFilterDiffPressure);"

J1939.PGN65271.REFRESH----------denotes you want to refresh PGN65271 data from vehicle. The relative interface method is "void processRefreshPGN65271_J1939(boolean Cmd_Success);"

J1939.PGN65271.GET_ALTERNATOR_VOLTAGE----------denotes you want to get Charging System Potential (Voltage). The relative interface method is "void processAlternatorVoltage_J1939(boolean Cmd_Success,float AlternatorVoltage);"

J1939.PGN65271.GET_ELECTRICAL_VOLTAGE----------denotes you want to get Battery Potential / Power Input 1. The relative interface method is "void processElectricalVoltage_J1939(boolean Cmd_Success,float ElectricalVoltage);"

J1939.PGN65271.GET_BATTERY_VOLTAGE----------denotes you want to get Keyswitch Battery Potential. The relative interface method is "void processBatteryVoltage_J1939(boolean Cmd_Success,float BatteryVoltage);"

J1939.PGN65272.REFRESH----------denotes you want to refresh PGN65272 data from vehicle. The relative interface method is "void processRefreshPGN65272_J1939(boolean Cmd_Success);"

J1939.PGN65272.GET_TRANSMISSION_OIL_LEVEL----------denotes you want to get Transmission Oil Level in percentage. The relative interface method is "void processTransmissionOilLevel_J1939(boolean Cmd_Success,float Percent);"

J1939.PGN65272.GET_TRANSMISSION_OIL_LEVEL_HIGH_LOW----------denotes you want to get Amount of current volume of transmission sump oil compared to

recommended volume. Positive values indicate overfill. Zero means the transmission fluild is filled to the recommended level. Unit is L. The relative interface method is "void processTransmissionOilLevelHighLow_J1939(boolean Cmd_Success,float HighLow);"

J1939.PGN65272.GET_TRANSMISSION_OIL_PRESSURE----------denotes you want to get Transmission Oil Pressure in kPa. The relative interface method is "void processTransmissionOilPressure_J1939(boolean Cmd_Success,float TransmissionOilPressure);"

J1939.PGN65272.GET_TRANSMISSION_OIL_TEMPERATURE----------denotes you want to get Transmission Oil Temperature in Celsius degree. The relative interface method is "void processTransmissionOilTemperature_J1939(boolean Cmd_Success,float Temperature);"

J1939.PGN65266.REFRESH----------denotes you want to refresh PGN65266 data from vehicle. The relative interface method is "void processRefreshPGN65266_J1939(boolean Cmd_Success);"

J1939.PGN65266.GET_FUEL_RATE----------denotes you want to get Engine Fuel Rate in L/H. The relative interface method is "void processFuelRate_J1939(boolean Cmd_Success,float FuelRate);"

J1939.PGN65266.GET_INSTANT_FUEL_ECONOMY----------denotes you want to get Engine Instantaneous Fuel Economy in Km/L. The relative interface method is "void processInstantFuelEconomy_J1939(boolean Cmd_Success,float InstantFuelEconomy);"

J1939.PGN65266.GET_AVG_FUEL_ECONOMY----------denotes you want to get Engine Average Fuel Economy in Km/L. The relative interface method is "void processAvgFuelEconomy_J1939(boolean Cmd_Success,float AvgFuelEconomy);"

J1939.PGN65266.GET_ENGINE_THROTTLE_POS----------denotes you want to get Engine Throttle Position in percentage. The relative interface method is "void processEngineThrottlePos_J1939(boolean Cmd_Success,float EngineThrottlePos);"

J1939.PGN65263.REFRESH----------denotes you want to refresh PGN65263 data from vehicle. The relative interface method is "void processRefreshPGN65263_J1939(boolean Cmd_Success);"

J1939.PGN65263.GET_FUEL_DELIVERY_PRESSURE----------denotes you want to get Engine Fuel Delivery Pressure in kPa. The relative interface method is "void processFuelDeliveryPressure_J1939(boolean Cmd_Success,float Pressure);"

J1939.PGN65263.GET_ENGINE_OIL_LEVEL----------denotes you want to get Engine Oil Level in percentage. The relative interface method is "void processEngineOilLevel_J1939(boolean Cmd_Success,float EngineOilLevel);"

J1939.PGN65263.GET_ENGINE_OIL_PRESSURE----------denotes you want to get Engine Oil Pressure in kPa. The relative interface method is "void processEngineOilPressure_J1939(boolean Cmd_Success,float EngineOilPressure);"

J1939.PGN65263.GET_ENGINE_COOLANT_PRESSURE----------denotes you want to get Engine Coolant Pressure in kPa. The relative interface method is "void processEngineCoolantPressure_J1939(boolean Cmd_Success,float EngineCoolantPressure);"

J1939.PGN65263.GET_ENGINE_COOLANT_LEVEL----------denotes you want to get Engine Coolant Level in percentage. The relative interface method is "void processEngineCoolantLevel_J1939(boolean Cmd_Success,float EngineCoolantLevel);"

J1939.PGN65253.REFRESH----------denotes you want to refresh PGN65253 data from vehicle. The relative interface method is "void processRefreshPGN65253_J1939(boolean Cmd_Success);"

J1939.PGN65253.GET_TOTAL_ENGINE_HOURS----------denotes you want to get Engine Total Hours of Operation. The relative interface method is "void processTotalEngineHours_J1939(boolean Cmd_Success,float TotalEngineHours);"

J1939.PGN65253.GET_TOTAL_ENGINE_REVOLUTIONS----------denotes you want to get Engine Total Revolutions. Unit is r. The relative interface method is "void processTotalEngineRevolutions_J1939(boolean Cmd_Success,float TotalEngineRevolutions);"

J1939.PGN65214.REFRESH----------denotes you want to refresh PGN65214 data from vehicle. The relative interface method is "void processRefreshPGN65214_J1939(boolean Cmd_Success);"

J1939.PGN65214.GET_RATED_ENGINE_SPEED----------denotes you want to get Engine Rated Speed in rpm. The relative interface method is "void processRatedEngineSpeed_J1939(boolean Cmd_Success,float RatedEngineSpeed);"

J1939.PGN65248.REFRESH----------denotes you want to refresh PGN65248 data from vehicle. The relative interface method is "void processRefreshPGN65248_J1939(boolean Cmd_Success);"

J1939.PGN65248.GET_TRIP_DISTANCE----------denotes you want to get Trip Distance in Km. The relative interface method is "void processTripDistance_J1939(boolean Cmd_Success,float TripDistance);"

J1939.PGN65248.GET_TOTAL_DISTANCE----------denotes you want to get Total Vehicle Distance in Km. The relative interface method is "void processTotalDistance_J1939(boolean Cmd_Success,float TotalDistance);"

J1939.PGN65276.REFRESH----------denotes you want to refresh PGN65276 data from vehicle. The relative interface method is "void processRefreshPGN65276_J1939(boolean Cmd_Success);"

J1939.PGN65276.GET_WASHER_FLUID_LEVEL----------denotes you want to get Washer Fluid Level in percentage. The relative interface method is "void processWasherFluidLevel_J1939(boolean Cmd_Success,float WasherFluidLevel);"

J1939.PGN65276.GET_FUEL_LEVEL1----------denotes you want to get Fuel Level 1 in percentage. The relative interface method is "void processFuelLevel1_J1939(boolean Cmd_Success,float FuelLevel1);"

J1939.PGN65276.GET_FUEL_LEVEL2----------denotes you want to get Fuel Level 2 in percentage. The relative interface method is "void processFuelLevel2_J1939(boolean Cmd_Success,float FuelLevel2);"

J1939.PGN65265.REFRESH----------denotes you want to refresh PGN65265 data from vehicle. The relative interface method is "void processRefreshPGN65265_J1939(boolean Cmd_Success);"

J1939.PGN65265.GET_WHEEL_BASED_VEHICLE_SPEED----------denotes you want to get Wheel-Based Vehicle Speed in km/h. The relative interface method is "void processWheelBasedVehicleSpeed_J1939(boolean Cmd_Success,float WheelBasedVehicleSpeed);"

J1939.PGN65265.GET_PARKING_BRAKE----------denotes you want to get Parking brake switch status: True/False. The relative interface method is "void processParkingBrake_J1939(boolean Cmd_Success,boolean ParkingBreakSet);"

J1939.PGN65265.GET_BRAKE----------denotes you want to get Brake switch status: True/False. Status "True" means Brake pedal depressed. Status "False" means Brake pedal released. The relative interface method is "void processBrake_J1939(boolean Cmd_Success,boolean BreakPedalDepressed);"

J1939.PGN57344.REFRESH----------denotes you want to refresh PGN57344 data from vehicle. The

relative interface method is "void processRefreshPGN57344_J1939(boolean Cmd_Success);"

J1939.PGN57344.GET_SEAT_BELT----------denotes you want to get status of Seat Belt Switch. The relative interface method is "void processSeatBelt_J1939(boolean Cmd_Success,boolean buckled);"

J1939.PGN64996.REFRESH----------denotes you want to refresh PGN64996 data from vehicle. The relative interface method is "void processRefreshPGN64996_J1939(boolean Cmd_Success);"

J1939.PGN64996.GET_PAYLOAD----------denotes you want to get Payload Percentage. The relative interface method is "void processPayLoad_J1939(boolean Cmd_Success, float PayLoad);"

J1939.PGN61445.REFRESH----------denotes you want to refresh PGN61445 data from vehicle. The relative interface method is "void processRefreshPGN61445_J1939(boolean Cmd_Success);"

J1939.PGN61445.GET_CURRENT_GEAR----------denotes you want to get Transmission Current Gear. The relative interface method is "void processCurrentGear_J1939(boolean Cmd_Success,float CurrentGear);"

J1939.PGN61445.GET_SELECTED_GEAR----------denotes you want to get Transmission Selected Gear. The relative interface method is "void processSelectedGear_J1939(boolean Cmd_Success,float SelectedGear);"

J1939.PGN65268.REFRESH----------denotes you want to refresh PGN65268 data from vehicle. The relative interface method is "void processRefreshPGN65268_J1939(boolean Cmd_Success);"

J1939.PGN65268.GET_TIRE_PRESSURE----------denotes you want to get tire pressure in kPa unit. The relative interface method is "void processTirePressure_J1939(boolean Cmd_Success,float TirePressure);"

J1939.PGN65268.GET_TIRE_TEMPERATURE----------denotes you want to get tire temperature in deg C unit. The relative interface method is "void processTireTemperature_J1939(boolean Cmd_Success,float Temperature);"

J1939.PGN65268.GET_TIRE_LOCATION----------denotes you want to identify which tire is associated with the parametric data in this PGN. The relative interface method is "void processTireLocation_J1939(boolean Cmd_Success,int Front2RearNumber,int Left2RighNumber); "

J1939.PGN65268.GET_TIRE_VALVE_PRESSURE_MONITOR----------denotes you want to get the pressure level of tire. The relative interface method is "void processTireValvePressureMonitor_J1939(boolean Cmd_Success,int

```
TireValvePressureMonitor);"
```

The above constants are set by "DFL168A. prefix", for example, "Message msg=new Message(); msg. what=DFL168A.J1939.PGN57344.GET_SEAT_BELT; "

4 For J1708/J1587, we have the following constants:

J1708.GET_AIR_PRESSURE----------denotes you want to get Gauge Pressure of air in system that utilizes compressed air to provide force between a lift axle and frame for purposes of lifting or lowering the axle, unit is kPa. The relative interface method is "`void processAirPressure_J1708(boolean Cmd_Success, float AirPressure);`"

J1708.GET_ENGINE_OIL_PRESSURE----------denotes you want to get Gauge pressure of oil in the engine lubrication system as provided by the oil pump, unit is kPa. The relative interface method is "`void processEngineOilPressure_J1708( boolean Cmd_Success,float EngineOilPressure);`"

J1708.GET_ENGINE_COOLANT_PRESSURE----------denotes you want to get Gauge pressure of liquid found in the engine cooling system, unit is kPa. The relative interface method is "`void processEngineCoolantPressure_J1708(boolean Cmd_Success,float EngineCoolantPressure);`"

J1708.GET_FUEL_LEVEL1----------denotes you want to get ratio of volume of fuel to the total volume of the primary fuel storage container. Unit is percentage. The relative interface method is "`void processFuelLevel1_J1708(boolean Cmd_Success,float FuelLevel1);`"

J1708.GET_FUEL_LEVEL2----------denotes you want to get ratio of volume of fuel to the total volume of the second fuel storage container. Unit is percentage. The relative interface method is "`void processFuelLevel2_J1708(boolean Cmd_Success,float FuelLevel2);`"

J1708.GET_BAROMETRIC_PRESSURE----------denotes you want to get absolute air pressure of the atmosphere in kPa. The relative interface method is "`void processBarometricPressure_J1708(boolean Cmd_Success,float Pressure);`"

J1708.GET_ENGINE_THROTTLE_POS----------denotes you want to get the position of the valve used to regulate the supply of a fluid, usually air or fuel/air mixture, to an engine. 0% represents no supply and 100% is full supply. The relative interface method is "`void processEngineThrottlePos_J1708(boolean Cmd_Success,float EngineThrottlePos);`"

J1708.GET_WHASHER_FLUID_LEVEL----------denotes you want to get ratio of volume of liquid to

total container volume of fluid reservoir in windshield wash system. The relative interface method is "`void processWasherFluidLevel_J1708( boolean Cmd_Success,float WasherFluidLevel);`"

J1708.GET_VEHICLE_SPEED----------denotes you want to get vehicle road speed in km/h. The relative interface method is "`void processVehicleSpeed_J1708(boolean Cmd_Success,float VehicleSpeed);`"

J1708.GET_ACCEL_PEDAL_POS1----------denotes you want to get ratio of actual accelerator pedal position to maximum pedal position. The relative interface method is "`void processAccelPedalPosi1_J1708(boolean Cmd_Success,float AccelPedalPosi1);`"

J1708.GET_ACCEL_PEDAL_POS2----------denotes you want to get ratio of actual accelerator pedal position to maximum pedal position. The relative interface method is "`void processAccelPedalPosi2_J1708(boolean Cmd_Success,float AccelPedalPosi2);`"

J1708.GET_ACCEL_PEDAL_POS3----------denotes you want to get ratio of actual accelerator pedal position to maximum pedal position. The relative interface method is "`void processAccelPedalPosi3_J1708(boolean Cmd_Success,float AccelPedalPosi3);`"

J1708.GET_ENGINE_LOAD----------denotes you want to get ratio of current output torque to maximum torque available at the current engine speed. The relative interface method is "`void processEngineLoad_J1708(boolean Cmd_Success, float Percent);`"

J1708.GET_ENGINE_OIL_LEVEL----------denotes you want to get ratio of current volume of engine sump oil to maximum required volume. The relative interface method is "`void processEngineOilLevel_J1708(boolean Cmd_Success,float EngineOilLevel); `"

J1708.GET_COOLANT_TEMPERATURE----------denotes you want to get the temperature of liquid found in engine cooling system in Celsius degree. The relative interface method is "`void processCoolantTemperature_J1708(boolean Cmd_Success, float Temperature);`"

J1708.GET_ENGINE_COOLANT_LEVEL----------denotes you want to get ratio of volume of liquid found in engine cooling system to total cooling system volume. The relative interface method is "`void processEngineCoolantLevel_J1708(boolean Cmd_Success,float EngineCoolantLevel); `"

J1708.GET_TRANSMISSION_OIL_LEVEL----------denotes you want to get ratio of volume of transmission sump oil to recommended volume. The relative interface method is "`void processTransmissionOilLevel_J1708(boolean Cmd_Success,`

float Percent); "

J1708.GET_TRANSMISSION_OIL_LEVEL_HIGH_LOW----------denotes you want to get amount of current volume of transmission sump oil compared to recommended volume. Unit is L. The relative interface method is "void processTransmissionOilLevelHighLow_J1708(boolean Cmd_Success, float HighLow);"

J1708.GET_TRANSMISSION_OIL_PRESSURE----------denotes you want to get gage pressure of lubrication fluid in transmission, measured after pump in kPa. The relative interface method is "void processTransmissionOilPressure_J1708(boolean Cmd_Success,float Pressure);"

J1708.GET_TRANSMISSION_OIL_TEMPERATURE----------denotes you want to get temperature of transmission lubricant in Celsius degree. The relative interface method is " void processTransmissionOilTemperature_J1708(boolean Cmd_Success,float Temperature);"

J1708.GET_POWER_SPECIFIC_INSTANT_FUEL_ECONOMY----------denotes you want to get instantaneous fuel economy of the engine, typically for off-highway equipment in kWh/L. The relative interface method is "void processPowerSpecificInstantFuelEconomy_J1708(boolean Cmd_Success,float Rate);"

J1708.GET_AVG_FUEL_RATE----------denotes you want to get continuous averaging fuel per hour per segment of engine operation in L/s. The relative interface method is "void processAvgFuelRate_J1708(boolean Cmd_Success,float FuelRate);"

J1708.GET_INSTANT_FUEL_ECONOMY----------denotes you want to get current fuel economy at current vehicle velocity in Km/L. The relative interface method is "void processInstantFuelEconomy_J1708(boolean Cmd_Success,float InstantFuelEconomy);"

J1708.GET_AVG_FUEL_ECONOMY----------denotes you want to get average of instantaneous fuel economy for that segment of vehicle operation of interest in Km/L. The relative interface method is "void processAvgFuelEconomy_J1708(boolean Cmd_Success,float AvgFuelEconomy);"

J1708.GET_ELECTRICAL_VOLTAGE----------denotes you want to get electrical potential measured at the input of the electronic control unit supplied through a switching device. The relative interface method is "void processElectricalVoltage_J1708(boolean Cmd_Success,float ElectricalVoltage); "

J1708.GET_RATED_ENGINE_POWER----------denotes you want to get net brake power that the

engine will deliver continuously, specified for a given application at a rated speed in KW. The relative interface method is "void processRatedEnginePower_J1708(boolean Cmd_Success,float Power);"

J1708.GET_BATTERY_VOLTAGE----------denotes you want to get measured electrical potential of the battery. The relative interface method is "void processBatteryVoltage_J1708(boolean Cmd_Success,float BatteryVoltage);"

J1708.GET_ALTERNATOR_VOLTAGE----------denotes you want to get measured electrical potential of the alternator. The relative interface method is "void processAlternatorVoltage_J1708(boolean Cmd_Success,float AlternatorVoltage);"

J1708.GET_AMBIENT_TEMPERATURE----------denotes you want to get temperature of air surrounding vehicle in Celsius degree. The relative interface method is "void processAmbientTemperature_J1708(boolean Cmd_Success,float AmbientTemperature);"

J1708.GET_CARGO_AMBIENT_TEMPERATURE----------denotes you want to get temperature of air inside vehicle container used to accommodate cargo in Celsius degree. The relative interface method is "void processCargoAmbientTemperature_J1708(boolean Cmd_Success, float CargoAmbientTemperature);"

J1708.GET_ROAD_TEMPERATURE----------denotes you want to get temperature of road surface over which vehicle is operating in Celsius degree. The relative interface method is "void processRoadTemperature_J1708(boolean Cmd_Success,float RoadTemperature);"

J1708.GET_CAB_INTERIOR_TEMPERATURE----------denotes you want to get temperature of air inside the part of the vehicle that encloses the driver and vehicle operating controls in Celsius degree. The relative interface method is "void processCabInteriorTemperature_J1708(boolean Cmd_Success,float CabInteriorTemperature);"

J1708.GET_INLET_TEMPERATURE----------denotes you want to get temperature of air entering vehicle air induction system in Celsius degree. The relative interface method is "void processInletTemperature_J1708(boolean Cmd_Success,float InletTemperature);"

J1708.GET_FUEL_TEMPERATURE----------denotes you want to get temperature of fuel entering injectors in Celsius degree. The relative interface method is "void processFuelTemperature_J1708(boolean Cmd_Success,float

FuelTemperature);"

J1708.GET_OIL_TEMPERATURE----------denotes you want to get temperature of engine lubricant in Celsius degree. The relative interface method is "void processOilTemperature_J1708(boolean Cmd_Success,float OilTemperature);"

J1708.GET_CARGO_WEIGHT----------denotes you want to get the force of gravity of freight carried in N. The relative interface method is "void processCargoWeight_J1708(boolean Cmd_Success,float CargoWeight);"

J1708.GET_ENGINE_TRIP_FUEL----------denotes you want to get the fuel consumed during all or part of a journey in L. The relative interface method is "void processEngineTripFuel_J1708(boolean Cmd_Success,float EngineTripFuel);"

J1708.GET_ENGINE_TOTAL_FUEL_USED----------denotes you want to get the accumulated amount of fuel used during vehicle operation in L. The relative interface method is "void processEngineTotalFuelUsed_J1708(boolean Cmd_Success, float EngineTotalFuelUsed);"

J1708.GET_FUEL_RATE----------denotes you want to get the amount of fuel consumed by engine per unit of time in L/s. The relative interface method is "void processFuelRate_J1708(boolean Cmd_Success,float FuelRate);"

J1708.GET_RATED_ENGINE_SPEED----------denotes you want to get the maximum governed rotational velocity of the engine crankshaft under full load conditions in rpm. The relative interface method is "void processRatedEngineSpeed_J1708(boolean Cmd_Success,float RatedEngineSpeed); "

J1708.GET_ENGINE_SPEED----------denotes you want to get the rotational velocity of crankshaft in rpm. The relative interface method is "void processEngineSpeed_J1708(boolean Cmd_Success,float EngineSpeed);"

J1708.GET_INTAKE_MANIFOLD_TEMPERATURE----------denotes you want to get the temperature of precombustion air found in intake manifold of engine air supply system in Celsius degree. The relative interface method is "void processIntakeManifoldTemperature_J1708(boolean Cmd_Success, float IntakeManifoldTemperature); "

J1708.GET_POWER_TAKEOFF_STATUS----------denotes you want to get the state of the system used to transmit engine power to auxiliary equipment. The relative interface method is "void processPowerTakeoffStatus_J1708(boolean Cmd_Success,boolean PTOModeActive, boolean ClutchSwitchOn, boolean BrakeSwitchOn, boolean AccelSwitchOn, boolean ResumeSwitchOn, boolean CoastSwitchOn, boolean SetSwitchOn,

boolean PTOControlSwitchOn);"

J1708.GET_TRIP_DISTANCE----------denotes you want to get the distance traveled during all or part of a journey in Km. The relative interface method is "void processTripDistance_J1708(boolean Cmd_Success,float TripDistance); "

J1708.GET_TOTAL_DISTANCE----------denotes you want to get the accumulated distance travelled by vehicle during its operation in Km. The relative interface method is " void processTotalDistance_J1708(boolean Cmd_Success,float TotalDistance);"

J1708.GET_TOTAL_ENGINE_HOURS----------denotes you want to get the accumulated time of operation of engine in hours. The relative interface method is "void processTotalEngineHours_J1708(boolean Cmd_Success,float TotalEngineHours);"

J1708.GET_TOTAL_ENGINE_REVOLUTIONS----------denotes you want to get the accumulated number of revolutions of engine crankshaft during its operation. The relative interface method is "void processTotalEngineRevolutions_J1708(boolean Cmd_Success,float TotalEngineRevolutions);"

J1708.GET_VIN----------denotes you want to get the Vehicle Identification Number (VIN) as assigned by the vehicle manufacturer. The relative interface method is "void processVIN_J1708(boolean Cmd_Success,String VIN);"

J1708.GET_DTC----------denotes you want to get the diagnostic code and occurrence count. The relative interface method is "void processDTC_J1708(boolean Cmd_Success,int DTC_Num,int MID,int PID_SID[],boolean IsPID[], int FMI[],boolean IsActive[],boolean OccurrenceExist[],int OccurrenceCount[]);"

J1708.CLEAR_DTC_PID----------denotes you want to clear diagnostic codes on the device with the given MID, PID. MID is set by message.arg1, PID is set by message.arg2. The relative interface method is "void processClearDTC_PID_J1708(boolean Cmd_Success);"

J1708.CLEAR_DTC_SID----------denotes you want to clear diagnostic codes on the device with the given MID, SID. MID is set by message.arg1, SID is set by message.arg2. The relative interface method is "void processClearDTC_SID_J1708(boolean Cmd_Success);"

J1708.GET_FAULT_DISCRIPTION_PID----------denotes you want to get DTC description string on the device with the given MID, PID, and FMI. FMI and MID are set by message.arg1, FMI is MSB of message.arg1, MID is LSB of message.arg1. PID is set by message.arg2. The relative interface method is "void

processFaultDescription_PID_J1708(boolean Cmd_Success,String FaultDescription);"

J1708.GET_FAULT_DISCRIPTION_SID----------denotes you want to get DTC description string on the device with the given MID, SID, and FMI. FMI and MID are set by message.arg1, FMI is MSB of message.arg1, MID is LSB of message.arg1. SID is set by message.arg2. The relative interface method is "void processFaultDescription_SID_J1708(boolean Cmd_Success,String FaultDescription);"

J1708.GET_PID_DISCRIPTION----------denotes you want to get PID description string on the device with the given MID. MID is set by message.arg1, PID is set by message.arg2. The relative interface method is "void processPIDDescription_J1708(boolean Cmd_Success,String PID_Description);"

J1708.GET_SID_DISCRIPTION----------denotes you want to get SID description string on the device with the given MID. MID is set by message.arg1, SID is set by message.arg2. The relative interface method is "void processSIDDescription_J1708(boolean Cmd_Success,String SID_Description);"

The above constants are set by "DFL168A. prefix", for example, "Message msg=new Message(); msg. what=DFL168A.J1708.GET_ENGINE_SPEED; "

5 For GPS, we have the following constants:

GPS.GET_GPS_INFO----------denotes you want to get GPS location and date and time information. The relative interface method is "void processGPSInfo(boolean Cmd_Success,float Latitude,float Longitude,float Speed,String Time, String Date);"

GPS.GET_ALTITUDE----------denotes you want to get altitude of vehicle location. The relative interface method is "void processGPSAltitude(boolean Cmd_Success,float Altitude);"

The above constants are set by "DFL168A. prefix", for example, "Message msg=new Message(); msg. what=DFL168A.GPS.GET_GPS_INFO; "

The only interface method which is not needed relative message is method "void processDFL168AStatus(boolean SleepWarning, boolean CancelWarning, boolean Sleep, boolean Wakeup);". However, it still need any other message to send by handler because IC found

sleep/sleep warning by IC other digital command running.

5.3 Methods

- DFL168A(Activity mActivity);
- DFL168A(Activity mActivity, byte currProtocol,int Timeout, int J1939Baudrate, int DEV_Baudrate, int DEV_timeout);
- DFL168A(Activity mActivity, BluetoothDevice mBluetoothDevice);
- DFL168A(Activity mActivity, BluetoothDevice mBluetoothDevice,byte currProtocol,int Timeout, int J1939Baudrate, int DEV_Baudrate, int DEV_timeout);
- void selectBlutoothDevice();
- void scanBlutoothDevice();
- boolean getBTSettingResult(int requestCode, int resultCode, Intent data);
- boolean getBTSettingResult(int requestCode, int resultCode, Intent data, boolean intrude, boolean Fast);
- boolean begin();
- boolean begin(boolean intrude, boolean Fast);
- void end();

5.3.1 Construction Function

- **1 DFL168A(Activity mActivity);**

Description

You must use this construction function to declare a DFL168A object. It will specify which activity defines DFL168A object, and we use protocol "AUTO_PROTOCOL" (ISO15765 automatic select). Protocol timeout is 500ms, GPS baud rate is 9600, and GPS timeout is 500ms

This construction is used when you use our Bluetooth device selection/scan graphic interface.

Syntax

DFL168A *YourObjectName= new* DFL168A(mActivity);

Parameters

mActivity: the first parameter, Activity type, We define DFL168A object inside an activity, so this parameter is "this", it denotes an activity instance.

Returns

Nothing

- **2 DFL168A(Activity mActivity, byte currProtocol,int Timeout, int J1939Baudrate, int DEV_Baudrate, int DEV_timeout);**

Description

You must use this construction function to declare a DFL168A object. It will specify which activity defines DFL168A object, which current protocol we use, what is Protocol timeout, what baud rate for J1939, what baud rate for DEV1 (in general, it is GPS) , what is DEV1 timeout

This construction is used when you use our Bluetooth device selection/scan graphic interface.

Syntax

DFL168A *YourObjectName= new* DFL168A(mActivity, currProtocol, Timeout, J1939Baudrate, DEV_Baudrate, DEV_timeout);

Parameters

mActivity: the 1st parameter, Activity type, We define DFL168A object inside an activity, so this parameter is "this", it denotes an activity instance.

currProtocol : the 2nd parameter, byte type. It tells DFL168A which protocol you select. It can be constant "DFL168A.AUTO_PROTOCOL", "DFL168A. ISO15765_11_500_PROTOCOL", "DFL168A.ISO15765_29_500_PROTOCOL" , "DFL168A.ISO15765_11_250_PROTOCOL", "DFL168A. ISO15765_29_250_PROTOCOL", "DFL168A.J1939_PROTOCOL", and "DFL168A. J1708_PROTOCOL"

Timeout: the 3rd parameter, int type. It actually sets up time out for DFL168A vehicle protocol. Its unit is ms. For example, we know one vehicle data will be broadcast every 2 seconds, so we can set up time out= 2000ms.

J1939Baudrate: The 4th parameter, int type. It sets up J1939 baud rate. Unit is "bps". If you didn't use J1939, this parameter can be any value.

DEV_Baudrate: the 5th parameter, int type. It is baud rate of DFL168A DEV1 serial port. Usually DEV1 serial port of DFL168A connects GPS module, so in this case, it will be GPS module baud rate. Of cause, DEV1 can be connected any other serial port device in the transparent mode of DFL168A. You can use message.what=DFL168A. BEGIN_TRANSPARENT_UART to enter transparent mode. And then you can use Uart object to access serial port device.

DEV_timeout: the 6th parameter, int type. It actually sets up time out (ms) for GPS module which is connected to DEV1 serial port of DFL168A. If GPS module send out NMEA-183 sentence every 1 seconds, you should let DEV_timeout=1000

Returns

Nothing

- **3 DFL168A(Activity mActivity, BluetoothDevice mBluetoothDevice);**

Description

You must use this construction function to declare a DFL168A object. It will specify which activity defines DFL168A object, which BluetoothDevice we use, and we use protocol "AUTO_PROTOCOL" (ISO15765 automatic select). Protocol timeout is 500ms, GPS baud rate is 9600, and GPS timeout is 500ms

This construction is used when you don't use our Bluetooth device selection/scan graphic interface. In this case, you must provide BluetoothDevice object mBluetoothDevice by yourself.
It is the same as construction function 1 except BluetoothDevice

Syntax

DFL168A *YourObjectName= new* DFL168A(mActivity, mBluetoothDevice);

Parameters

mActivity: the1st parameter, Activity type, We define DFL168A object inside an activity, so this parameter is "this", it denotes an activity instance.

mBluetoothDevice: the 2nd parameter, BluetoothDevice class type, it denotes a BluetoothDevice instance.

Returns

Nothing

- **4 DFL168A(Activity mActivity, BluetoothDevice mBluetoothDevice,byte currProtocol,int Timeout, int J1939Baudrate, int DEV_Baudrate, int DEV_timeout);**

Description

You must use this construction function to declare a DFL168A object. It will specify which activity defines DFL168A object, which BluetoothDevice we use, which current protocol we use, what is Protocol timeout, what baud rate for J1939, what baud rate for DEV1 (in general, it is GPS) , what is DEV1 timeout

This construction is used when you don't use our Bluetooth device selection/scan graphic interface. In this case, you must provide BluetoothDevice object mBluetoothDevice by yourself.

It is the same as construction function 2 except BluetoothDevice

Syntax

DFL168A *YourObjectName= new* DFL168A(mActivity, mBluetoothDevice, currProtocol, Timeout, J1939Baudrate, DEV_Baudrate, DEV_timeout);

Parameters

mActivity: the 1st parameter, Activity type, We define DFL168A object inside an activity, so this parameter is "this", it denotes an activity instance.

mBluetoothDevice: the 2nd parameter, BluetoothDevice class type, it denotes a BluetoothDevice instance.

currProtocol : the 3rd parameter, byte type. It tells DFL168A which protocol you select. It can be constant "DFL168A.AUTO_PROTOCOL", "DFL168A.ISO15765_11_500_PROTOCOL", "DFL168A.ISO15765_29_500_PROTOCOL" , "DFL168A.ISO15765_11_250_PROTOCOL", "DFL168A.ISO15765_29_250_PROTOCOL", "DFL168A.J1939_PROTOCOL", and "DFL168A.J1708_PROTOCOL"

Timeout: the 4th parameter, int type. It actually sets up time out for DFL168A vehicle protocol. Its unit is ms. For example, we know one vehicle data will be broadcast every 2 seconds, so we can set up time out= 2000ms.

J1939Baudrate: The 5th parameter, int type. It sets up J1939 baud rate. Unit is "bps". If you didn't use J1939, this parameter can be any value.

DEV_Baudrate: the 6th parameter, int type. It is baud rate of DFL168A DEV1 serial port. Usually DEV1 serial port of DFL168A connects GPS module, so in this case, it will be GPS module baud rate. Of cause, DEV1 can be connected any other serial port device in the transparent mode of DFL168A. You can use message.what=DFL168A.BEGIN_TRANSPARENT_UART to enter transparent mode. And then you can use Uart object to access serial port device.

DEV_timeout: the 7th parameter, int type. It actually sets up time out (ms) for GPS module which is connected to DEV1 serial port of DFL168A. If GPS module send out NMEA-183 sentence every 1 seconds, you should let DEV_timeout=1000

Returns

Nothing

5.3.2 selectBlutoothDevice Method

- **void selectBlutoothDevice();**

Description

selectBlutoothDevice() will display graphic interface for user to select Bluetooth Device.
It is only used when you use our Bluetooth device selection/scan graphic interface.

Syntax

DFL168A.selectBlutoothDevice();

Parameters

Nothing

Returns

Nothing

5.3.3 scanBlutoothDevice Method

- **void scanBlutoothDevice();**

Description

scanBlutoothDevice() will display graphic interface for user to scan and select Bluetooth Device.
Bluetooth devices list contains previous Bluetooth devices and new found Bluetooth devices.
It is only used when you use our Bluetooth device selection/scan graphic interface.

Syntax

DFL168A.scanBlutoothDevice();

Parameters

Nothing

Returns

Nothing

5.3.4 getBTSettingResult Method

- **1 boolean getBTSettingResult(int requestCode, int resultCode, Intent data);**

Description

It is only used when you use our Bluetooth device selection/scan graphic interface.
Firstly, you must override method of "`void onActivityResult(int requestCode, int resultCode, Intent data)`" by "Alt+Insert" shortcut in your activity which defines DFL168A object. And in this method, you must call getBTSettingResult() method.

getBTSettingResult() will get Bluetooth setting result and call begin() method with intrude=true and fast =false automatically. So you don't need call begin method again.

Syntax

DFL168A.getBTSettingResult(requestCode, resultCode, data);

Parameters

requestCode: the 1st parameter, int type. It comes from `onActivityResult method 1st parameter.`

resultCode: the 2nd parameter, int type. It comes from `onActivityResult method 2nd parameter.`

data: the 3rd parameter, `Intent` type. It comes from `onActivityResult method 3rd parameter.`

Returns

bool type, true: success in initialization of DFL168A IC

false: failure in initialization of DFL168A IC . The reason can be known by reading variable "`ErrorCode`"

`ErrorCode has the following constant value:`

ERROR_NO_UART ----------value is 0, denotes that Bluetooth cannot become valid uart

ERROR_THREAD_SLEEP----------value is 1, denotes that sleep delay exception occurs.

ERROR_INVALID_DFL168A_IC---------value is 2, denotes that invalid DFL168A IC is used.

ERROR_AT_COMMAND----------value is 3, denotes that AT Command cannot run correctly

ERROR_WRONG_PROTOCOL----------value is 4, denotes that vehicle does not support current selected protocol.

- **2 boolean getBTSettingResult(int requestCode, int resultCode, Intent data, boolean intrude, boolean Fast);**

Description

It is only used when you use our Bluetooth device selection/scan graphic interface. Firstly, you must override method of "`void onActivityResult(int requestCode, int resultCode, Intent data)`" by "Alt+Insert" shortcut in your activity which defines DFL168A object. And in this method, you must call getBTSettingResult() method.

getBTSettingResult() will get Bluetooth setting result and call begin(intrude,fast) method automatically. So you don't need call begin method again.

Syntax

DFL168A.getBTSettingResult(requestCode, resultCode, data, intrude, fast);

Parameters

requestCode: the 1st parameter, int type. It comes from `onActivityResult method 1st parameter.`

resultCode: the 2nd parameter, int type. It comes from `onActivityResult method 2nd parameter.`

data: the 3rd parameter, `Intent` type. It comes from `onActivityResult method 3rd parameter.`

intrude: the 4th parameter, boolean type. It will give it to the 1st parameter of begin method.

fast: the 5th parameter, boolean type. It will give it to the 2nd parameter of begin method.

Returns

bool type, true: success in initialization of DFL168A IC

false: failure in initialization of DFL168A IC . The reason can be known by reading variable "`ErrorCode`"

`ErrorCode has the following constant value:`

ERROR_NO_UART ----------value is 0, denotes that Bluetooth cannot become valid uart

ERROR_THREAD_SLEEP----------value is 1, denotes that sleep delay exception occurs.

ERROR_INVALID_DFL168A_IC---------value is 2, denotes that invalid DFL168A IC is used.

ERROR_AT_COMMAND----------value is 3, denotes that AT Command cannot run correctly

ERROR_WRONG_PROTOCOL----------value is 4, denotes that vehicle does not support current selected protocol.

5.3.5 begin Method

- **1 boolean begin(boolean intrude,boolean Fast);**

Description

DFL168A.begin(intrude,Fast) will config DFL168A IC and check whether DFL168A Configuration is OK. If configuration succeed, it will return true, otherwise return false. This method will run as long as 5 seconds for setting DFL168A IC. And progress dialog will display how long will be done.

Syntax

DFL168A.begin(intrude,Fast);

Parameters

intrude: the first parameter, boolean type. It denotes DFL168A will send request to vehicle when its value is true. And DFL168A won't send request to vehicle for J1708/J1939 protocol when its value is false, so DFL168A only uses broadcast of vehicle data in this situation for J1708/ J1939 protocol.

Fast: the second parameter, boolean type. It won't send any config data to DFL168A if its value is true. So in this situation, you should setup DFL168A by yourself via hyper-terminal. Library will send all config data to DFL168A if its value is false. You don't need take care of anything. Library takes care of everything for you.

Returns

boolean type, true: success in initialization of DFL168A IC
false: failure in initialization of DFL168A IC . The reason can be known by reading variable "ErrorCode"

ErrorCode has the following constant value:

ERROR_NO_UART ----------value is 0, denotes that Bluetooth cannot become valid uart
ERROR_THREAD_SLEEP----------value is 1, denotes that sleep delay exception occurs.
ERROR_INVALID_DFL168A_IC---------value is 2, denotes that invalid DFL168A IC is used.
ERROR_AT_COMMAND----------value is 3, denotes that AT Command cannot run correctly
ERROR_WRONG_PROTOCOL----------value is 4, denotes that vehicle does not support current selected protocol.

- **2 boolean begin();**

Description

DFL168A.begin() will config DFL168A IC and check whether DFL168A Configuration is OK. If configuration succeed, it will return true, otherwise return false. This method will run as long as 5 seconds for setting DFL168A IC. And progress dialog will display how long will be done.
It is the same as DFL168A.begin(intrude,Fast) when intrude=true and Fast=false

Syntax

DFL168A.begin();

Parameters

None

Returns

boolean type, true: success in initialization of DFL168A IC
false: failure in initialization of DFL168A IC . The reason can be known by reading variable "ErrorCode"

ErrorCode has the following constant value:

ERROR_NO_UART ----------value is 0, denotes that Bluetooth cannot become valid uart
ERROR_THREAD_SLEEP----------value is 1, denotes that sleep delay exception occurs.
ERROR_INVALID_DFL168A_IC---------value is 2, denotes that invalid DFL168A IC is used.
ERROR_AT_COMMAND----------value is 3, denotes that AT Command cannot run correctly
ERROR_WRONG_PROTOCOL----------value is 4, denotes that vehicle does not support current selected protocol.

5.3.6 end Method

- **void end();**

Description

DFL168A.end() will release some resources such as Bluetooth serial port which is used by DFL168A, and You can use DFL168A.begin(intrude,Fast) again only after you run this method.

Syntax

DFL168A.end();

Parameters

Nothing

Returns

Nothing

6 DFL168A_ResultProcess Interface

Please read DFL168A Constant part about all methods meaning of this interface.
We give you this interface source code below:

```
public interface DFL168A_ResultProcess {
     void processDFL168AStatus(boolean SleepWarning, boolean CancelWarning,
     boolean Sleep, boolean Wakeup);
     void processDIN(boolean Cmd_Success,int port, boolean Value);
     void processOnewireID(boolean Cmd_Success, int[] ID);
     void processDOUT( int port, boolean Value);
     void processAnalogInput(boolean Cmd_Success,float AnalogInputValue);
     void processBeginTransparentSerial();
     void processEndTransparentSerial(boolean Cmd_Success);
     void processSetExitTransparentKey(boolean Cmd_Success);
     void processSetSleepDelay(boolean Cmd_Success);

     //the followings are ISO15765
     void processCoolantTemperature(boolean Cmd_Success,float temperature);
     void processEngineSpeed(boolean Cmd_Success, float EngineSpeed);
     void processVehicleSpeed(boolean Cmd_Success,float VehicleSpeed);
     void processIntakeManifoldPressure(boolean Cmd_Success,float
     IntakeManifoldPressure);
     void processFuelSystemStatus(boolean Cmd_Success,boolean A_Openloop,
     boolean A_Closedloop,boolean A_OpenloopByDriving_Con,boolean
     A_OpenloopByFault,boolean A_ClosedloopButFault,boolean B_Openloop,
     boolean B_Closedloop,boolean B_OpenloopByDriving_Con,boolean
     B_OpenloopByFault,boolean B_ClosedloopButFault);
     void processCalculatedLoadValue(boolean Cmd_Success,float
     CalculatedLoad);
     void processShortTermFuelTrimBank13(boolean Cmd_Success,float Bank1,float
     Bank3);
     void processLongTermFuelTrimBank13(boolean Cmd_Success,float Bank1,float
     Bank3);
     void processShortTermFuelTrimBank24(boolean Cmd_Success,float Bank2,float
```

```
Bank4);
void processLongTermFuelTrimBank24(boolean Cmd_Success,float Bank2,float
Bank4);
void processIgnitionTimingAdvance(boolean Cmd_Success,float Angle);
void processIntakeAirTemperature(boolean Cmd_Success,float temperature);

void processAirFlowRateFrmMAF(boolean Cmd_Success,float FlowRate);
void processAbsThrottlePosition(boolean Cmd_Success,float Percent);
void processOxygenSensorLocation(boolean Cmd_Success,boolean
Bank1_Sensor1Present, boolean Bank1_Sensor2Present, boolean
Bank1_Sensor3Present,boolean Bank1_Sensor4Present, boolean
Bank3_Sensor1Present,boolean Bank3_Sensor2Present,boolean
Bank2_Sensor1Present, boolean Bank2_Sensor2Present, boolean
Bank2_Sensor3Present,boolean Bank2_Sensor4Present, boolean
Bank4_Sensor1Present,boolean Bank4_Sensor2Present);
void processBank1OSensor1Voltage(boolean Cmd_Success,float OutVoltage);
void processBank1OSensor2Voltage(boolean Cmd_Success,float OutVoltage);
void processBank1OSensor3Voltage(boolean Cmd_Success,float OutVoltage);
void processBank1OSensor4Voltage(boolean Cmd_Success,float OutVoltage);
void processBank2OSensor1Voltage(boolean Cmd_Success,float OutVoltage);
void processBank2OSensor2Voltage(boolean Cmd_Success,float OutVoltage);
void processBank2OSensor3Voltage(boolean Cmd_Success,float OutVoltage);
void processBank2OSensor4Voltage(boolean Cmd_Success,float OutVoltage);
void processBank3OSensor1Voltage(boolean Cmd_Success,float OutVoltage);
void processBank3OSensor2Voltage(boolean Cmd_Success,float OutVoltage);
void processBank4OSensor1Voltage(boolean Cmd_Success,float OutVoltage);
void processBank4OSensor2Voltage(boolean Cmd_Success,float OutVoltage);
void processOBDCertified(boolean Cmd_Success,String OBD);
void processTimeSinceEngineStart(boolean Cmd_Success,float TotalTime);
void processDistanceTraveledMIL(boolean Cmd_Success,float Distance);
void processFuelRailPressure(boolean Cmd_Success,float Pressure);
void processFuelLevelInput(boolean Cmd_Success,float Percent);
void processDistanceTraveledSinceDTC_Clear(boolean Cmd_Success,float
Distance);
void processBarometricPressure(boolean Cmd_Success,float Pressure);
void processControlModuleVoltage(boolean Cmd_Success,float Voltage);
void processRelativeThrottlePos(boolean Cmd_Success,float Percent);
void processAmbientTemp(boolean Cmd_Success,float AmbientTemp);
void processCommandedThrottleActuatorControl(boolean Cmd_Success,float
Percent);
void processEngineRunTimeMIL(boolean Cmd_Success,float TotalTime);
void processEngineRunTimeSinceDTC_Clear(boolean Cmd_Success,float
TotalTime);
void processTypeOfFuelUsedCurrently(boolean Cmd_Success,String
FuelType);
void processRelativeAcceleratorPedalPosition(boolean Cmd_Success,float
Percent);
void processHybridBatteryPackRemainingLife(boolean Cmd_Success,float
Percent);
void processEngineOilTemperature(boolean Cmd_Success,float Temperature);
void processFuelRate(boolean Cmd_Success,float FuelRate);
void processActualEngineTorque(boolean Cmd_Success,float
ActualEngineTorque);
void processMILStatus(boolean Cmd_Success,boolean MIL_IS_ON);
void processEngineRunTime(boolean Cmd_Success,float TotalEngineRunTime,
```

```
float TotalIdleRunTime,float TotalRunTimeWithPTOActive);
void processVIN(boolean Cmd_Success,String VIN);
void processDTC(boolean Cmd_Success,int DTC_Num,String DTC[]);
void processClearDTC(boolean Cmd_Success);

//the followings are J1708
void processAirPressure_J1708(boolean Cmd_Success,float AirPressure);
void processEngineOilPressure_J1708(boolean Cmd_Success,float
EngineOilPressure);
void processEngineCoolantPressure_J1708(boolean Cmd_Success,float
EngineCoolantPressure);
void processFuelLevel1_J1708(boolean Cmd_Success,float FuelLevel1);
void processFuelLevel2_J1708(boolean Cmd_Success,float FuelLevel2);
void processBarometricPressure_J1708(boolean Cmd_Success,float Pressure);

void processEngineThrottlePos_J1708(boolean Cmd_Success,float
EngineThrottlePos);
void processWasherFluidLevel_J1708(boolean Cmd_Success,float
WasherFluidLevel);
void processVehicleSpeed_J1708(boolean Cmd_Success,float VehicleSpeed);

void processAccelPedalPosi1_J1708(boolean Cmd_Success,float
AccelPedalPosi1);
void processAccelPedalPosi2_J1708(boolean Cmd_Success,float
AccelPedalPosi2);
void processAccelPedalPosi3_J1708(boolean Cmd_Success,float
AccelPedalPosi3);
void processEngineLoad_J1708(boolean Cmd_Success,float Percent);
void processEngineOilLevel_J1708(boolean Cmd_Success,float
EngineOilLevel);
void processCoolantTemperature_J1708(boolean Cmd_Success,float
Temperature);
void processEngineCoolantLevel_J1708(boolean Cmd_Success,float
EngineCoolantLevel);
void processTransmissionOilLevel_J1708(boolean Cmd_Success,float
Percent);
void processTransmissionOilLevelHighLow_J1708(boolean Cmd_Success,float
HighLow);
void processTransmissionOilPressure_J1708(boolean Cmd_Success,float
Pressure);
void processTransmissionOilTemperature_J1708(boolean Cmd_Success,float
Temperature);
void processPowerSpecificInstantFuelEconomy_J1708(boolean Cmd_Success,
float Rate);
void processAvgFuelRate_J1708(boolean Cmd_Success,float FuelRate);
void processInstantFuelEconomy_J1708(boolean Cmd_Success,float
InstantFuelEconomy);
void processAvgFuelEconomy_J1708(boolean Cmd_Success,float
AvgFuelEconomy);
void processElectricalVoltage_J1708(boolean Cmd_Success,float
ElectricalVoltage);
void processRatedEnginePower_J1708(boolean Cmd_Success,float Power);
void processBatteryVoltage_J1708(boolean Cmd_Success,float
BatteryVoltage);
void processAlternatorVoltage_J1708(boolean Cmd_Success,float
```

```
AlternatorVoltage);
void processAmbientTemperature_J1708(boolean Cmd_Success,float
AmbientTemperature);
void processCargoAmbientTemperature_J1708(boolean Cmd_Success,float
CargoAmbientTemperature);
void processRoadTemperature_J1708(boolean Cmd_Success,float
RoadTemperature);
void processCabInteriorTemperature_J1708(boolean Cmd_Success,float
CabInteriorTemperature);
void processInletTemperature_J1708(boolean Cmd_Success,float
InletTemperature);
void processFuelTemperature_J1708(boolean Cmd_Success,float
FuelTemperature);
void processOilTemperature_J1708(boolean Cmd_Success,float
OilTemperature);
void processCargoWeight_J1708(boolean Cmd_Success,float CargoWeight);
void processEngineTripFuel_J1708(boolean Cmd_Success,float
EngineTripFuel);
void processEngineTotalFuelUsed_J1708(boolean Cmd_Success,float
EngineTotalFuelUsed);
void processFuelRate_J1708(boolean Cmd_Success,float FuelRate);
void processRatedEngineSpeed_J1708(boolean Cmd_Success,float
RatedEngineSpeed);
void processEngineSpeed_J1708(boolean Cmd_Success,float EngineSpeed);
void processIntakeManifoldTemperature_J1708(boolean Cmd_Success,float
IntakeManifoldTemperature);
void processPowerTakeoffStatus_J1708(boolean Cmd_Success,boolean
PTOModeActive, boolean ClutchSwitchOn, boolean BrakeSwitchOn, boolean
AccelSwitchOn, boolean ResumeSwitchOn, boolean CoastSwitchOn, boolean
SetSwitchOn, boolean PTOControlSwitchOn);
void processTripDistance_J1708(boolean Cmd_Success,float TripDistance);

void processTotalDistance_J1708(boolean Cmd_Success,float TotalDistance);

void processTotalEngineHours_J1708(boolean Cmd_Success,float
TotalEngineHours);
void processTotalEngineRevolutions_J1708(boolean Cmd_Success,float
TotalEngineRevolutions);
void processVIN_J1708(boolean Cmd_Success,String VIN);
void processDTC_J1708(boolean Cmd_Success,int DTC_Num,int MID,int
PID_SID[],boolean IsPID[], int FMI[],boolean IsActive[],boolean
OccurrenceExist[],int OccurrenceCount[]);
void processClearDTC_PID_J1708(boolean Cmd_Success);
void processClearDTC_SID_J1708(boolean Cmd_Success);
void processFaultDescription_PID_J1708(boolean Cmd_Success,String
FaultDescription);
void processFaultDescription_SID_J1708(boolean Cmd_Success,String
FaultDescription);
void processPIDDescription_J1708(boolean Cmd_Success,String
PID_Description);
void processSIDDescription_J1708(boolean Cmd_Success,String
SID_Description);


//the followings are J1939
```

```
void processVIN_J1939(boolean Cmd_Success,String VIN);
void processDTC_J1939(boolean Cmd_Success,int DTC_Num,int SPN[], int
FMI[],int CM[], int OC[]);
void processClearDTC_J1939(boolean Cmd_Success);

void processRefreshPGN65267_J1939(boolean Cmd_Success);
void processLatitude_J1939(boolean Cmd_Success,float Latitude);
void processLongitude_J1939(boolean Cmd_Success,float Longitude);

void processRefreshPGN65262_J1939(boolean Cmd_Success);
void processCoolantTemperature_J1939(boolean Cmd_Success,float
Temperature);
void processFuelTemperature_J1939(boolean Cmd_Success,float Temperature);
void processOilTemperature_J1939(boolean Cmd_Success,float Temperature);

void processRefreshPGN65256_J1939(boolean Cmd_Success);
void processAltitude_J1939(boolean Cmd_Success,float Altitude);
void processNavBasedSpeed_J1939(boolean Cmd_Success,float Speed);

void processRefreshPGN65269_J1939(boolean Cmd_Success);
void processBarometricPressure_J1939(boolean Cmd_Success,float Pressure);
void processAmbientTemperature_J1939(boolean Cmd_Success,float
Temperature);
void processInletTemperature_J1939(boolean Cmd_Success,float
Temperature);
void processtRoadTemperature_J1939(boolean Cmd_Success,float
Temperature);
void processCabInteriorTemperature_J1939(boolean Cmd_Success,float
Temperature);

void processRefreshPGN65257_J1939(boolean Cmd_Success);
void processEngineTripFuel_J1939(boolean Cmd_Success,float
EngineTripFuel);
void processEngineTotalFuelUsed_J1939(boolean Cmd_Success,float
EngineTotalFuelUsed);

void processRefreshPGN61444_J1939(boolean Cmd_Success);
void processActualEngineTorque_J1939(boolean Cmd_Success,float
ActualEngineTorque);
void processEngineSpeed_J1939(boolean Cmd_Success,float EngineSpeed);

void processRefreshPGN61443_J1939(boolean Cmd_Success);
void processAccelPedalPosi1_J1939(boolean Cmd_Success,float
AccelPedalPosi1);
void processAccelPedalPosi2_J1939(boolean Cmd_Success,float
AccelPedalPosi2);
void processEnginePerLoadAtCurrSpeed_J1939(boolean Cmd_Success,float
EnginePerLoadAtCurrSpeed);

void processRefreshPGN65270_J1939(boolean Cmd_Success);
void processIntakeManifoldPressure_J1939(boolean Cmd_Success,float
Pressure);
void processIntakeManifoldTemperature_J1939(boolean Cmd_Success,float
Temperature);
void processEngineAirInletPressure_J1939(boolean Cmd_Success,float
```

```
EngineAirInletPressure);
void processEngineExhaustGasTemperature_J1939(boolean Cmd_Success,float
Temperature);
void processEngineAirFilterDiffPressure_J1939(boolean Cmd_Success,float
EngineAirFilterDiffPressure);

void processRefreshPGN65271_J1939(boolean Cmd_Success);
void processAlternatorVoltage_J1939(boolean Cmd_Success,float
AlternatorVoltage);
void processElectricalVoltage_J1939(boolean Cmd_Success,float
ElectricalVoltage);
void processBatteryVoltage_J1939(boolean Cmd_Success,float
BatteryVoltage);

void processRefreshPGN65272_J1939(boolean Cmd_Success);
void processTransmissionOilLevel_J1939(boolean Cmd_Success,float
Percent);
void processTransmissionOilLevelHighLow_J1939(boolean Cmd_Success,float
HighLow);
void processTransmissionOilPressure_J1939(boolean Cmd_Success,float
TransmissionOilPressure);
void processTransmissionOilTemperature_J1939(boolean Cmd_Success,float
Temperature);

void processRefreshPGN65266_J1939(boolean Cmd_Success);
void processFuelRate_J1939(boolean Cmd_Success,float FuelRate);
void processInstantFuelEconomy_J1939(boolean Cmd_Success,float
InstantFuelEconomy);
void processAvgFuelEconomy_J1939(boolean Cmd_Success,float
AvgFuelEconomy);
void processEngineThrottlePos_J1939(boolean Cmd_Success,float
EngineThrottlePos);

void processRefreshPGN65263_J1939(boolean Cmd_Success);
void processFuelDeliveryPressure_J1939(boolean Cmd_Success,float
Pressure);
void processEngineOilLevel_J1939(boolean Cmd_Success,float
EngineOilLevel);
void processEngineOilPressure_J1939(boolean Cmd_Success,float
EngineOilPressure);
void processEngineCoolantPressure_J1939(boolean Cmd_Success,float
EngineCoolantPressure);
void processEngineCoolantLevel_J1939(boolean Cmd_Success,float
EngineCoolantLevel);

void processRefreshPGN65253_J1939(boolean Cmd_Success);
void processTotalEngineHours_J1939(boolean Cmd_Success,float
TotalEngineHours);
void processTotalEngineRevolutions_J1939(boolean Cmd_Success,float
TotalEngineRevolutions);

void processRefreshPGN65214_J1939(boolean Cmd_Success);
void processRatedEngineSpeed_J1939(boolean Cmd_Success,float
RatedEngineSpeed);
```

```
        void processRefreshPGN65248_J1939(boolean Cmd_Success);
        void processTripDistance_J1939(boolean Cmd_Success,float TripDistance);
        void processTotalDistance_J1939(boolean Cmd_Success,float TotalDistance);

        void processRefreshPGN65276_J1939(boolean Cmd_Success);
        void processWasherFluidLevel_J1939(boolean Cmd_Success,float
        WasherFluidLevel);
        void processFuelLevel1_J1939(boolean Cmd_Success,float FuelLevel1);
        void processFuelLevel2_J1939(boolean Cmd_Success,float FuelLevel2);

        void processRefreshPGN65265_J1939(boolean Cmd_Success);
        void processWheelBasedVehicleSpeed_J1939(boolean Cmd_Success,float
        WheelBasedVehicleSpeed);
        void processParkingBrake_J1939(boolean Cmd_Success,boolean
        ParkingBreakSet);
        void processBrake_J1939(boolean Cmd_Success,boolean BreakPedalDepressed);

        void processRefreshPGN57344_J1939(boolean Cmd_Success);
        void processSeatBelt_J1939(boolean Cmd_Success,boolean buckled);

        void processRefreshPGN64996_J1939(boolean Cmd_Success);
        void processPayLoad_J1939(boolean Cmd_Success,float PayLoad);

        void processRefreshPGN61445_J1939(boolean Cmd_Success);
        void processCurrentGear_J1939(boolean Cmd_Success,float CurrentGear);
        void processSelectedGear_J1939(boolean Cmd_Success,float SelectedGear);

        void processRefreshPGN65268_J1939(boolean Cmd_Success);
        void processTireTemperature_J1939(boolean Cmd_Success,float Temperature);
        void processTirePressure_J1939(boolean Cmd_Success,float TirePressure);
        void processTireLocation_J1939(boolean Cmd_Success,int Front2RearNumber,
        int Left2RighNumber);
        void processTireValvePressureMonitor_J1939(boolean Cmd_Success,int
        TireValvePressureMonitor);

        //the followings are for GPS
        void processGPSInfo(boolean Cmd_Success,float Latitude,float Longitude,
        float Speed,String Time, String Date);
        void processGPSAltitude(boolean Cmd_Success,float Altitude);

    }
```

7 Examples

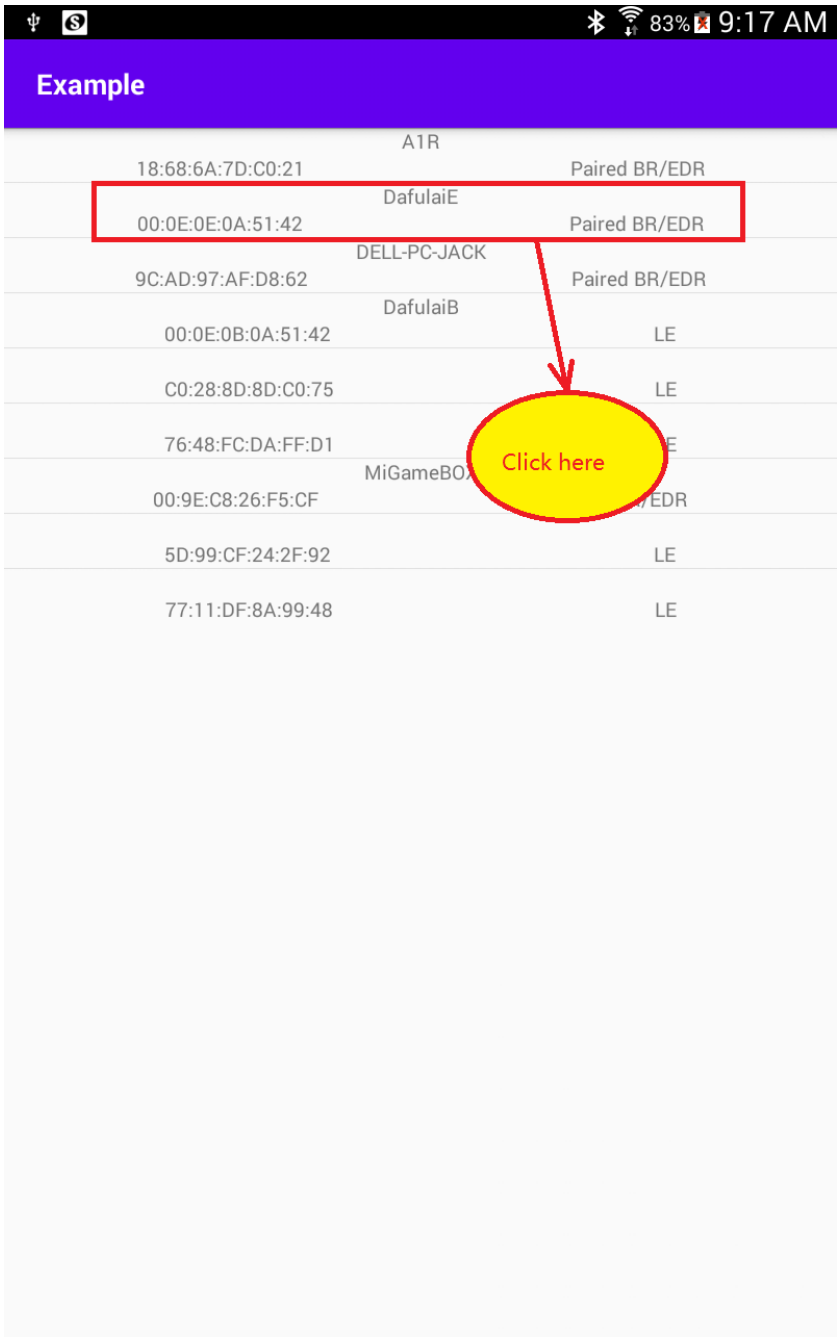
The following simple example will tell you how to get J1939 and GPS result.

For J1939 protocol with baud rate of 250k bit/sec, we will get VIN, DTC, and clear DTC when DTC exists, and "Accelerator Pedal position 1" , and "Engine Percent Load at Current Speed"

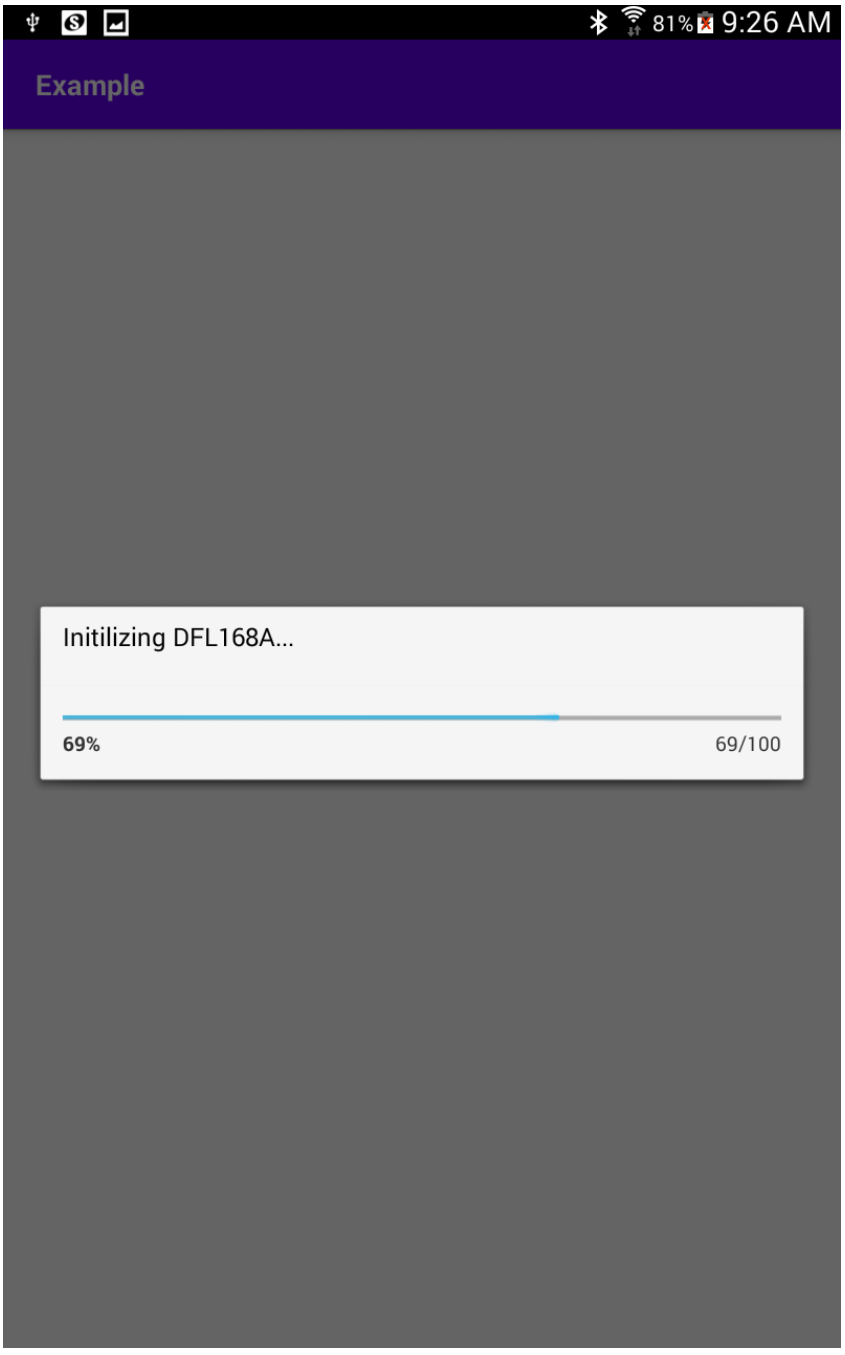
For GPS baud rate 9600, we will get Latitude, Longitude, Speed, Time, and Date.

We used Bluetooth select interface provided by DFL168A library.

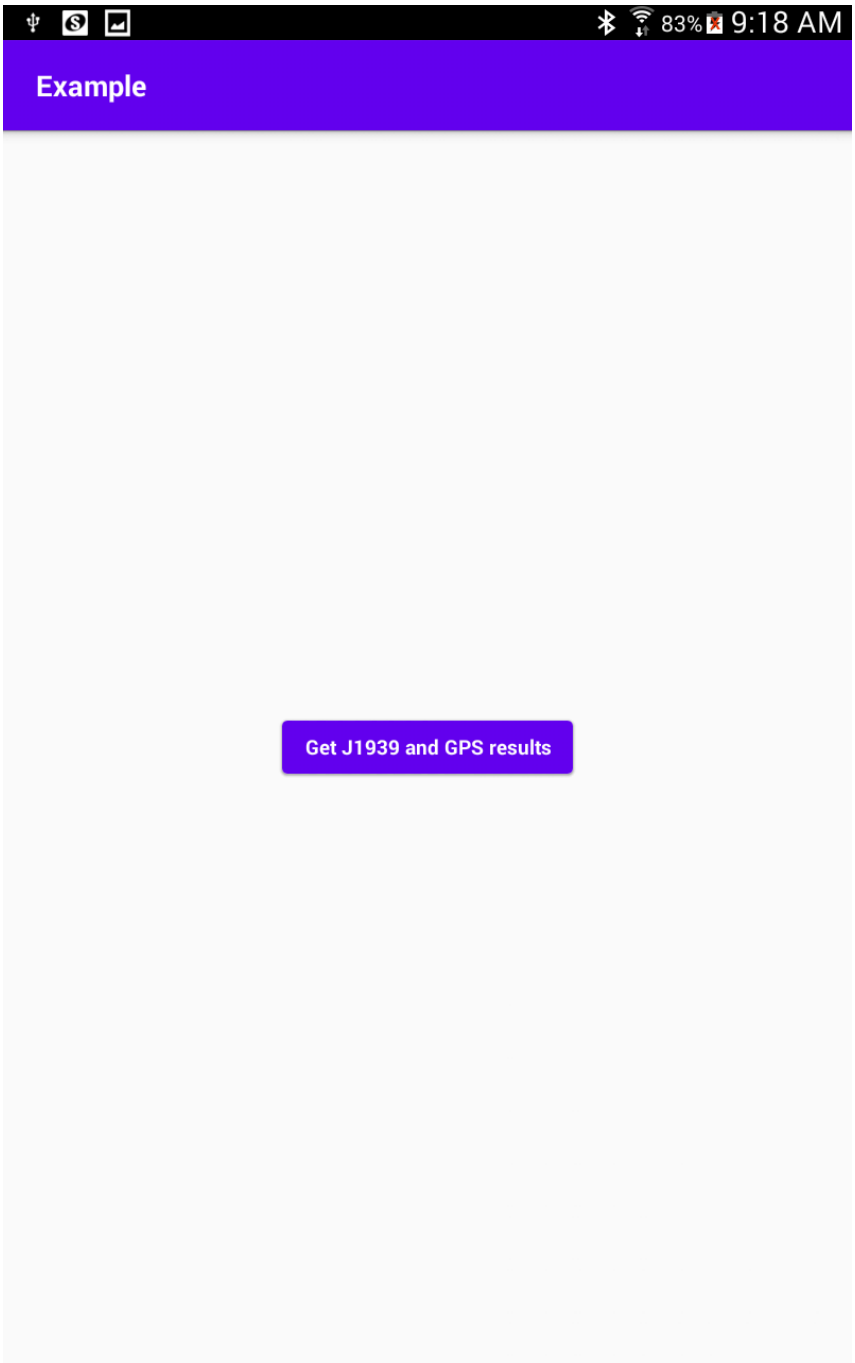
When run this example, it opens Bluetooth select graphic window, See screenshot below:



And after you choose correct Bluetooth device which is DFL168A connected as screenshot above (It may be not the item in the screenshot, you have to choose as your environment), you will see screenshot below:

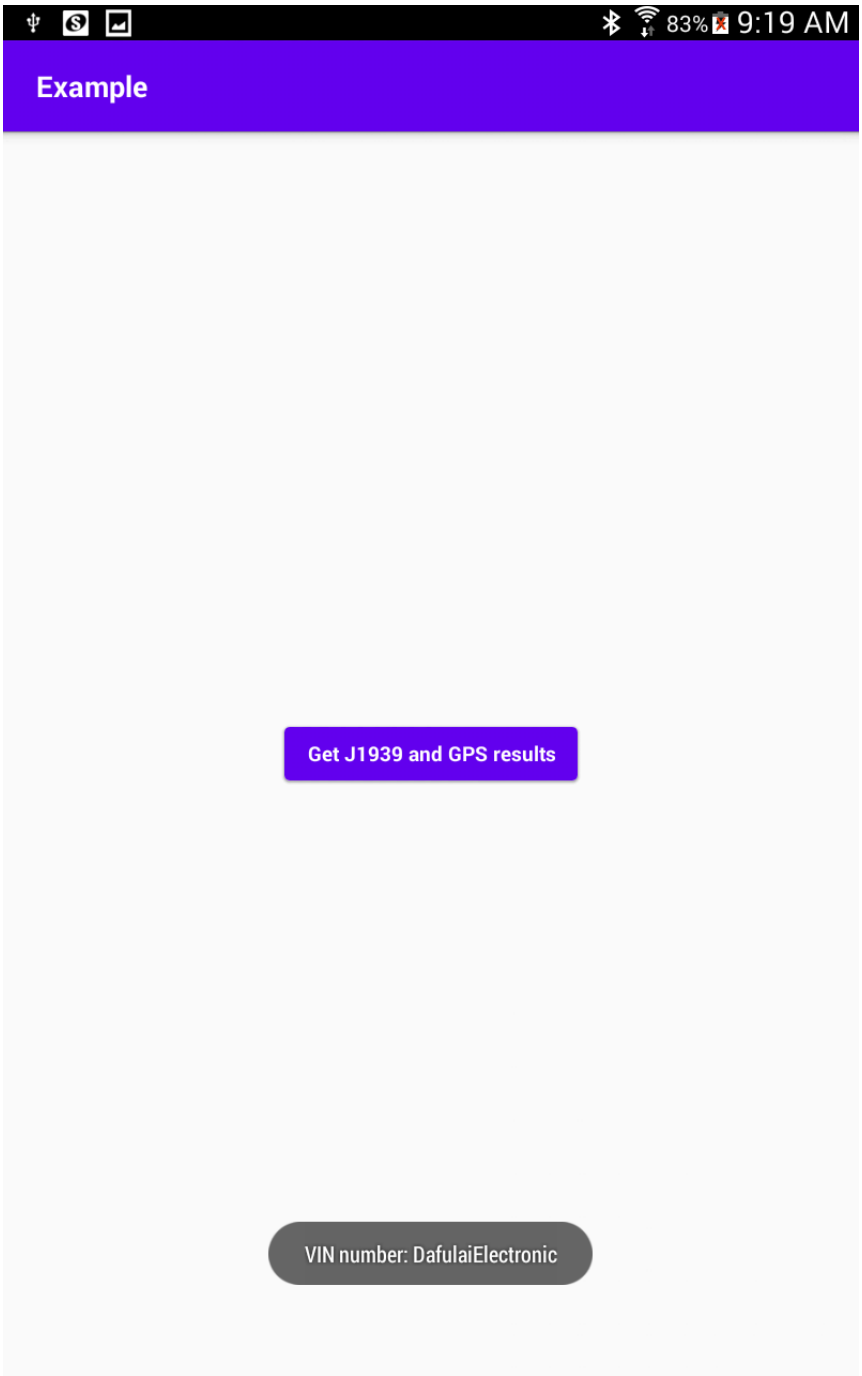


After initialization of DFL168A, you will see your main activity with "Get J1939 and GPS results" button as screenshot below:



. If DFL168A IC is OK, The button "Get J1939 and GPS results" will be enabled, and you can click this button, and you will get all results by notification (Toast).

Of cause, your smart phone must enable App Notifications. Please see screenshot below:



The source code of example can be downloaded from http://dafulaielectronics.com/DFL168A_Android_Example.zip Let's explain how to setup our example software.

Firstly, you must set up project with empty activity (or basic activity, or whatever other activity). Android Studio will set up MainActivity.java and relative layout activity_main.xml.

Then, please follow How to install library in Android Studio (chapter 3) to install DFL168A library.

Now change Layout file activity_main.xml as below:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android
```

```
="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btnGet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Get J1939 and GPS results"
        android:textAllCaps="false"
        android:enabled="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Change java file MainActivity.java as below:

```
package com.dafulaielectronics.example;

import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.dafulaielectronics.dfl168a.DFL168A;

public class MainActivity extends AppCompatActivity  {
    private Button mBtn;
    private DFL168A mDFL168A;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mDFL168A=new DFL168A(this,DFL168A.J1939_PROTOCOL,1000,250000,9600,1000
);
        mDFL168A.scanBlutoothDevice();
        mBtn=(Button) findViewById(R.id.btnGet);
        mBtn.setOnClickListener(new View.OnClickListener() {

            @Override
```

```
            public void onClick(View view) {
               Message msg=new Message();
                msg.what=DFL168A.J1939.GET_VIN;
                mDFL168A.handler.sendMessage(msg);
            }
        });

    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (mDFL168A.getBTSettingResult(requestCode,resultCode,data)) {
            mBtn.setEnabled(true);
        }
        else {
            mBtn.setEnabled(false);
        }
    }
}
```

Our MainActivity.java must implement DFL168A_ResultProcess interface. So change `MainActivity class to implement` DFL168A_ResultProcess interface like below:

```
package com.dafulaielectronics.example;

import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.dafulaielectronics.dfl168a.DFL168A;
import com.dafulaielectronics.dfl168a.DFL168A_ResultProcess;
public class MainActivity extends AppCompatActivity  {
    private Button mBtn;
    private DFL168A mDFL168A;
    @Override
    protected void onCreate(Bundle savedInstanceState) implements
DFL168A_ResultProcess {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mDFL168A=new DFL168A(this,DFL168A.J1939_PROTOCOL,1000,250000,9600,1000
);
        mDFL168A.scanBlutoothDevice();
        mBtn=(Button) findViewById(R.id.btnGet);
        mBtn.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View view) {
                Message msg=new Message();
                 msg.what=DFL168A.J1939.GET_VIN;
                 mDFL168A.handler.sendMessage(msg);
            }
        });

    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (mDFL168A.getBTSettingResult(requestCode,resultCode,data)) {
            mBtn.setEnabled(true);
        }
        else {
            mBtn.setEnabled(false);
        }
    }
}
```

In the MainActivity.java editor window, press short cut "ALT+Insert" to pop up a menu and click menu item "Implement Methods...", pop up an dialog "Select Methods to implement", Android Studio select all methods automatically for you, you just click OK button.
And write code as below:

```
package com.dafulaielectronics.example;

import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.dafulaielectronics.dfl168a.DFL168A;
import com.dafulaielectronics.dfl168a.DFL168A_ResultProcess;
public class MainActivity extends AppCompatActivity  {
    private Button mBtn;
    private DFL168A mDFL168A;
    @Override
    protected void onCreate(Bundle savedInstanceState) implements
DFL168A_ResultProcess {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mDFL168A=new DFL168A(this,DFL168A.J1939_PROTOCOL,1000,250000,9600,1000
);
        mDFL168A.scanBlutoothDevice();
        mBtn=(Button) findViewById(R.id.btnGet);
        mBtn.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View view) {
               Message msg=new Message();
                msg.what=DFL168A.J1939.GET_VIN;
                mDFL168A.handler.sendMessage(msg);
            }
        });

    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (mDFL168A.getBTSettingResult(requestCode,resultCode,data)) {
            mBtn.setEnabled(true);
        }
        else {
            mBtn.setEnabled(false);
        }
    }

@Override
public void processVIN_J1939(boolean b, String s) {
     if (b) {
         Toast.makeText(this,"VIN number: " +s,Toast.LENGTH_SHORT).show();
     }
     else {
         Toast.makeText(this,"Fail in VIN" ,Toast.LENGTH_SHORT).show();
     }
     Message msg=new Message();
     msg.what=DFL168A.J1939.GET_DTC;
     msg.arg1=4;
     mDFL168A.handler.sendMessage(msg);
}

@Override
public void processDTC_J1939(boolean b, int DTC_Num, int[] SPN, int[] FMI, int
[] CM, int[] OC) {
    if (b) {
        Toast.makeText(this,"DTC quantity: " +DTC_Num,Toast.LENGTH_SHORT
).show();
        int i;
        for (i=0;i<DTC_Num;i++) {
            Toast.makeText(this,"SPN(" +i+")="+SPN[i],Toast.LENGTH_SHORT
).show();
```

```
            Toast.makeText(this,"FMI(" +i+")="+FMI[i],Toast.LENGTH_SHORT
).show();
            Toast.makeText(this,"CM(" +i+")="+CM[i],Toast.LENGTH_SHORT).show();
            Toast.makeText(this,"OC(" +i+")="+OC[i],Toast.LENGTH_SHORT).show();
        }
    }
    else {
        Toast.makeText(this,"Fail in getting DTC" ,Toast.LENGTH_SHORT).show();
    }
    Message msg=new Message();
    if (DTC_Num>0)
    {
        msg.what=DFL168A.J1939.CLEAR_DTC;
        mDFL168A.handler.sendMessage(msg);
    }
    else {
        msg.what=DFL168A.J1939.PGN61443.REFRESH;
        mDFL168A.handler.sendMessage(msg);
    }
}

@Override
public void processClearDTC_J1939(boolean b) {
    if (b) {
        Toast.makeText(this,"Clear DTC successfuly " ,Toast.LENGTH_SHORT
).show();
    }
    else {
        Toast.makeText(this,"Fail in clear DTC" ,Toast.LENGTH_SHORT).show();
    }
    Message msg=new Message();
    msg.what=DFL168A.J1939.PGN61443.REFRESH;

    mDFL168A.handler.sendMessage(msg);
}

@Override
public void processRefreshPGN61443_J1939(boolean b) {
    if (b) {
        Toast.makeText(this,"Refresh PGN61443 successfuly " ,Toast.LENGTH_SHORT
).show();
        Message msg=new Message();
        msg.what=DFL168A.J1939.PGN61443.GET_ACCEL_PEDAL_POSI1;
        mDFL168A.handler.sendMessage(msg);
    }
    else {
        Toast.makeText(this,"Fail in refresh PGN61443" ,Toast.LENGTH_SHORT
).show();
    }

}

@Override
public void processAccelPedalPosi1_J1939(boolean b, float v) {
    if (b) {
```

```
        Toast.makeText(this,"accelerator Pedal 1: "+ v+"%" ,Toast.LENGTH_SHORT
).show();
    }
    else {
        Toast.makeText(this,"Fail in getting accelerator Pedal 1" ,Toast.
LENGTH_SHORT).show();
    }
    Message msg=new Message();
    msg.what=DFL168A.J1939.PGN61443.GET_ENGINE_PERLOAD_AT_CURRENT_SPEED;
    mDFL168A.handler.sendMessage(msg);
}

@Override
public void processAccelPedalPosi2_J1939(boolean b, float v) {

}

@Override
public void processEnginePerLoadAtCurrSpeed_J1939(boolean b, float v) {
    if (b) {
        Toast.makeText(this,"Engine percent load at current speed: "+ v+"%" ,
Toast.LENGTH_SHORT).show();
    }
    else {
        Toast.makeText(this,"Fail in getting engine percent load at current
speed" ,Toast.LENGTH_SHORT).show();
    }
}

@Override
public void processDFL168AStatus(boolean SleepWarning, boolean CancelWarning,
boolean Sleep, boolean Wakeup) {
     if (SleepWarning) {
         Toast.makeText(this,"DFL168A will sleep after a moment" ,Toast.
LENGTH_SHORT).show();
     }
     if (CancelWarning) {
        Toast.makeText(this,"DFL168A sleep is cancelled" ,Toast.LENGTH_SHORT
).show();
     }
     if (Sleep) {
        Toast.makeText(this,"DFL168A shutdown" ,Toast.LENGTH_SHORT).show();
     }
     if (Wakeup) {
        Toast.makeText(this,"DFL168A Wakeup" ,Toast.LENGTH_SHORT).show();
     }
}

//We ignore the other methods of interface, but you should keep empty methods
of interface
//...........

}
```

At last you can make project.