astra telematics

Application Note: SDK Reading FMS PGNs

Scope
AT240, AT110

Overview
This document gives an example of reading FMS PGNs that are not supported by default in the SDK. It also gives an example of adding the data to the packets in the reporting protocol.

Related Documents
The following documents are recommended reading to accompany this document:

- SDK Users Guide

Standards

FMS Standard
You can download the FMS Standard from the FMS-Standard website after registering here http://www.fms-standard.com/down_load/download_access.htm.

Source Files
The two files that need to be edited are j1939.h and j1939.c and will be in the sdk folder under inc and src respectively. These are already in the CrossWorks Solution in the Project Explorer.

In j1939.h add an entry to end of list for #define X_J1939_FMS_PARAMETER. Here we are adding PGN 0xFE56 for Diesel Exhaust Fluid Tank 1 Information so the list now looks as follows:

```
#define X_J1939_FMS_PARAMETER \
  X(CRUISE_CTRL_SPEED, 0xFEF1) \
  .
  .
  .
  X(HIGH_RES_FUEL_CONSUMPTION, 0xFD09) \
  X(DIESEL_EXHAUST_TANK_1, 0xFE56)
```

Note the entry before the last one now ends with a '\' character. Only the last entry should be without the '\' character.

Then in j1939.c in function j1939_fms_parse() add a case statement

```
    case J1939_FMS_PGN_DIESEL_EXHAUST_TANK_1:
      // Diesel Exhaust Fluid Tank 1 Level is in Data Byte 1
      // 0-250 represents 0 to 100%
      val8 = (uint8_t)(can_rx_msg->data_0_3 & 0xFF);

      // Store data in fms_report structure
      break;
```

The two variables can_rx_msg->data_0_3 and can_rx_msg->data_4_7 are both 32 bit values. The least significant byte of can_rx_msg->data_0_3 is Data Byte 1. The most significant byte of can_rx_msg->data_0_3 is Data Byte 4. Similarly, the least significant byte of can_rx_msg->data_4_7 is Data Byte 5 and the most significant byte of can_rx_msg->data_4_7 is Data Byte 8.

Reading data of different sizes
Examples of reading 8, 16 and 32 bit values are shown below, but also see reading values from existing PGNs in j1939.c

```
val8 = (uint8_t)(can_rx_msg->data_0_3 & 0xFF); // 8 bit value in Data Byte 1
val8 = (uint8_t)(can_rx_msg->data_0_3 >> 8);   // 8 bit value in Data Byte 2
val8 = (uint8_t)(can_rx_msg->data_0_3 >> 16);  // 8 bit value in Data Byte 3
val8 = (uint8_t)(can_rx_msg->data_0_3 >> 24);  // 8 bit value in Data Byte 4
val8 = (uint8_t)(can_rx_msg->data_4_7 & 0xFF); // 8 bit value in Data Byte 5
val8 = (uint8_t)(can_rx_msg->data_4_7 >> 16);  // 8 bit value in Data Byte 7

val16 = (uint16_t)(can_rx_msg->data_0_3 >> 8); // 16 bit value in Data Bytes 2-3
val16 = (uint16_t)(can_rx_msg->data_4_7 >> 16);// 16 bit value in Data Bytes 7-8

val32 = can_rx_msg->data_0_3;                  // 32 bit value in Data Bytes 1-4
val32 = can_rx_msg->data_4_7;                  // 32 bit value in Data Bytes 5-8
```

You should add member variables to the fms_report structure to store the extra data that you read. This structure is of type canbus_report_t and is defined in canbus.h. For example

```
      uint8_t diesel_exhaust_level;
```

Reporting the data
Define a new protocol element in the enumeration in comms.h. For example

```
    PROT_ELEM_CANBUS_EXHAUST_DIESEL,
```

In the comms.c module you will need to add this to the protocol_V_basic_map if you want it in every protocol V report. If you want it in start and stop reports add it before PROT_ELEM_STOP_ONLY in protocol_fms_obd_extra_map. If you want it in stop reports only add it after PROT_ELEM_STOP_ONLY in protocol_fms_obd_extra_map.

Then in the function build_packet() in comms.c add a case statement

```
    case case  PROT_ELEM_CANBUS_EXHAUST_DIESEL:
      msg[iLen++] = Report->canbus.adblue_level;
      break;
```

and decode it from the report packet appropriately.