DEEP THOUGHT
SYSTEMS PVT. LTD

DTS-OBD STACK USER MANUAL V1.0

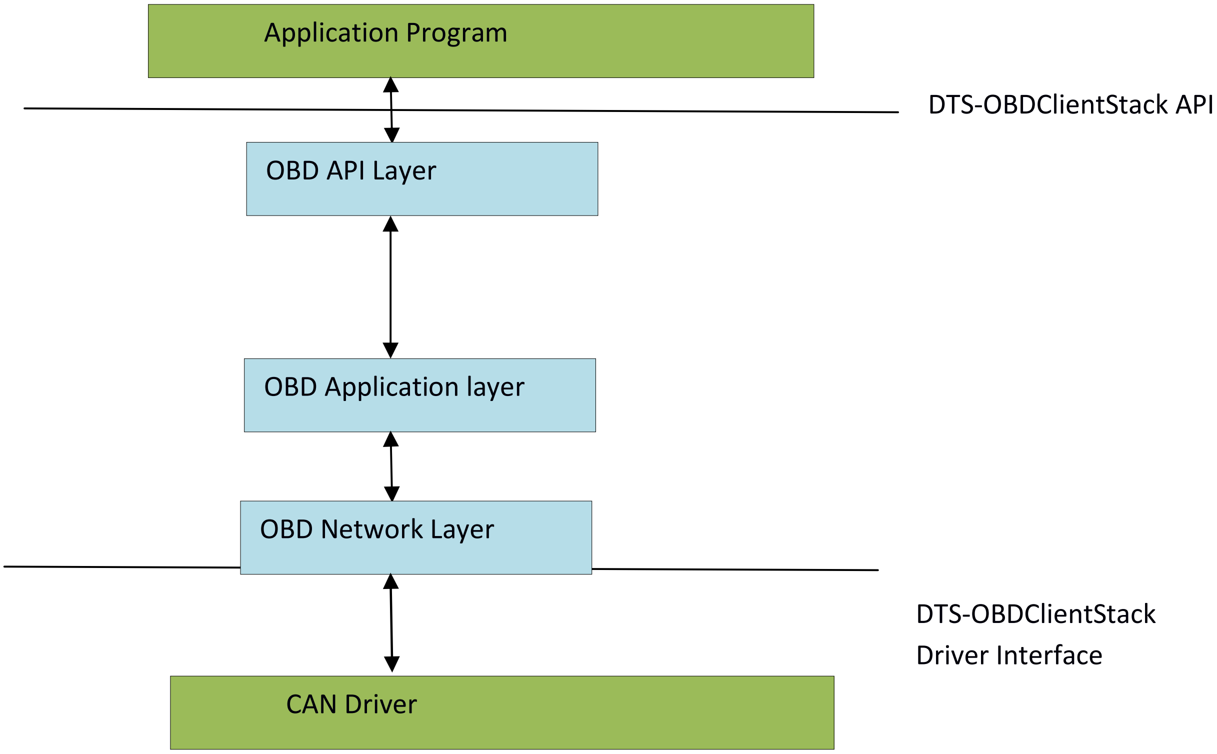
Contents

Introduction

This user manual is aimed at the users of DTS_OBDClientStack, which means programmers who try to integrate our stack with their hardware. Hence this document explains the stack architecture, the configuration options and how to use them, Function reference section, Macro references and sample code demonstrating the usage examples. DTS_OBDClientStack is a ISO15765 compliant portable OBD stack written in C.

DTS_OBD Client Stack Architecture

This stack is designed for single threaded operation. Lowest layer of the stack is CAN driver which provides interfaces for CAN controller operation. OBD network layer as per the ISO15765-2 standards is layered on top of the CAN driver which handles both single frame and multi frame messages.. OBD application layer conforms to the ISO15765-4 standard and OBD API layer is the top most layer which exposes various OBD APIs to the application program.



CAN Driver is platform specific and should implement the DTS-OBD Client Stack Driver Interface

DTS-OBD Client Stack External Interfaces

Application interface

OBDStack.c file implements the external application interface and it is exposed through the OBDStack.h file

OBD Layer API is used by the application program

The functions used in OBDStack file are listed below

INT16 InitOBD(void);

INT16 OBDSendRequest(INT16 nMode, UCHAR8 uchParam);

INT16 OBDReadReply(UCHAR8** ppBuf, INT16* pnLen);

void GetVIN(UCHAR8* VinBuf, UCHAR8* puchBuf);

UINT16 ParseDTC(UCHAR8 DtcBuf[][5], UCHAR8* puchBuf);

OBD Layer API Function Reference

1. INT16 OBDInit(void)

   This function is used to Initialize the OBD driver.
   parameters: Nil
   Return Value: Function returns OBD_INIT_SUCCESS if success
   or OBD_NO_PROTOCOL if failed.
   Data type : INT16

2. INT16 OBDSendRequest(INT16 nMode, UCHAR8 uchParam)

   This function is used to request OBD parameter value from the OBD ECU.
   Parameters:

   a. nMode (IN) :
      Data type : INT16

OBD request mode to be used. Can be one of the following values.

MODE_ENGINE_PARAMS
MODE_GET_DTC
MODE_CLEAR_DTC
MODE_PENDING_DTC
MODE_VEHICLE_INFO

b. uchParam : OBD parameter requested. Please refer to Appendix A for list of supported OBD parameters

Return Value: Function returns OBD_SUCCESS if successful or one of the following error codes.

OBD_NOTSUPPORTED_PARAM : if the requested parameter is not supported by the vehicle
OBD_REQ_PENDING : if there is already an outstanding OBD Request.

3. INT16 OBDReadReply(UCHAR8** ppBuf, INT16* pnLen)

This function is used to check size and determine single frame or multi frame, and read the data from CAN network
Parameters:
a) ppBuf(OU): obd buffer output
Data type: UCHAR8
Read Reply data from the CAN network

b) pnLen(IN): INT16
Data length

Return Value: nRet
Data type : INT16
if data available process OBD_SUCCESS
else OBD_READ_PENDING and it still waiting for the reply

4. void GetVIN(UCHAR8* VinBuf, UCHAR8* puchBuf)

This function is used to get the Vehicle Identification number Mode 09 and PID 02 to the VinBuf parameter.

Parameters :

a) VinBuf(OU)

Data type : UCHAR8

Vehicle information are stored in VIN BUF

b) puchBuf(IN)

Data Type: UCHAR8

passing the buffer pointer obtained from previous function.

Return Value : Nil

5. UINT16 ParseDTC(UCHAR8 DtcBuf[][5], UCHAR8* puchBuf)

This function returns a list of the DTCs that have been set. Mode 3 . No PID required. Each trouble code requires 2 bytes to describe.The first character in the trouble code is determined by the first two bits in the first byte:

Parameters :

a) DtcBuf[][5](OU):

Data Type : UCHAR8

DTC Buffer output provides DTC codes in the 2 dimensional array.

b) puchBuf(IN):

Data Type: UCHAR8

passing the buffer pointer obtained from previous function.

Returns :

uchDTCCount :

Data type: INT16

This function returns the no. of DTC

OBD Layer API Macro reference

1. GET_LOAD_PCT(uchLoadPCT,puchBuf)

This macro is used extract the Engine Load percentage from the OBD buffer.

Parameters:

uchLoadPCT (OUT)

Type : UCHAR8

Extracted Engine Load percentage value is given in this parameter

puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

2. GET_ECT(nEct,puchBuf)

This macro is used to extract Engine Coolant Temperature.

Parameters:

a. nEct (OUT)

Type : UCHAR8

Extracted Engine Coolant Temperature value is given in this parameter

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

3. GET_MAP(uchMap,puchBuf)

This macro is used extract the Intake Manifold Absolute Pressure.

Parameters:

a. uchMap (OUT)

Type : UCHAR8

Extracted Intake Manifold Absolute Pressure value is given in this parameter

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

4. GET_RPM(flRpm,puchBuf)

This macro is used extract the Engine RPM.

Parameters:

a. flRpm (OUT)

Type : UCHAR8

Extracted Engine RPM value is given in this parameter

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

5. GET_SPEED(uchSpeed,puchBuf)

This macro is used extract the Vehicle Speed Sensor Value.

Parameters:

a. uchSpeed, (OUT)

Type : UCHAR8

Extracted Vehicle Speed Sensor value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

6. GET_SPARKADV(flSparkadv,puchBuf)

This macro is used to extract the Ignition Timing Advance for #1 Cylinder
Parameters:

a. flSparkadv (OUT)

Type : FLOAT32

Extracted ignition Timing Advance for #1 Cylinder value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

7. GET_IAT(nIat,puchBuf)

This macro is used to extract the Intake Air Temperature
Parameters:

a. nIat(OUT)

Type : UCHAR8

Extracted intake Air Temperature value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

8. GET_MAF(flMaf,puchBuf)

This macro is used to extract the Air flow rate from Mass Air Flow Sensor.
Parameters:

a. flMaf(OUT)

Type : FLOAT32

Extracted Air flow rate from Mass Air Flow Sensor value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

9. GET_TP(uchTp,puchBuf)

This macro is used to extract the Absoulte Throttle Pressure.

Parameters:

a. uchTp(OUT)

Type : UCHAR8

Extracted Absoulte Throttle Pressure value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

10. GET_RUNTM(nRuntm,puchBuf)

This macro is used to extract the Time Since Engine Start.

Parameters:

a. nRuntm(OUT)

Type : UCHAR8

Extracted the Time Since Engine Start value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

11. GET_MIL_DIST(unMilDist,puchBuf)

This macro is used to extract the Distance traveled while MIL is activated.

Parameters:

a. unMilDist(OUT)

Type : UCHAR8

Extracted the Distance traveled value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

12. GET_EGR_PCT(uchEgrPct,puchBuf)

This macro is used to extract the commandedexhaust gas recirculation .

Parameters:

a. uchEgrPct(OUT)

Type : UCHAR8

Extracted the commanded EGR value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

13. GET_EVAP_PCT(uchEvapPct,puchBuf)

This macro is used to extract the commanded evaporative purge.

Parameters:

a. uchEvapPct(OUT)

Type :   UCHAR8

Extracted the commanded evaporative purge value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

14. GET_FLI(uchFli,puchBuf)

This macro is used to extract the Fuel Tank Level Input. Parameters:

a. uchFli(OUT)

Type :   UCHAR8

Extracted the Fuel Level Input value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

15. GET_WARM_UPS(uchWarmUps,puchBuf)

This macro is used to extract the Number of warm-ups since DTC cleared.

Parameters:

a. uchWarmUp(OUT)

Type :   UCHAR8

Extracted the Number of warm-ups is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]

Buffer containing the OBD data.

16. GET_CLR_DST(unClrDst,puchBuf)

This macro is used to extract the Distance traveled since DTC cleared.
Parameters:

a. unClrDst(OUT)
Type : UCHAR8
Extracted the Distance traveled since DTC cleared is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

17. GET_BARO(uchBaro,puchBuf)

This macro is used to extract the Barometric Pressure.
Parameters:

a. unClrDst(OUT)
Type : UCHAR8
Extracted the Barometric Pressure value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

18. GET_CATEMP11(nCattemp11,puchBuf)

This macro is used to extract Catalyst Temperature Bank 1, Sensor 1.
Parameters:

a. nCattemp11(OUT)
Type : UCHAR8
Extracted the Catalyst Temperature Bank 1, Sensor 1 value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

19. GET_CATEMP21(nCattemp21,puchBuf)

This macro is used to extract Catalyst Temperature Bank 2, Sensor 1.
Parameters:

a. nCattemp21(OUT)
Type : UCHAR8
Extracted Catalyst Temperature Bank 2, Sensor 1 value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

20. GET_CATEMP12(nCattemp12,puchBuf)

This macro is used to extract Catalyst Temperature Bank 1, Sensor 2.
Parameters:

a. nCattemp12(OUT)
Type : UCHAR8

Extracted Catalyst Temperature Bank 1, Sensor 2 value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

21. GET_CATEMP22(nCattemp22,puchBuf)

This macro is used to extract Catalyst Temperature Bank 2, Sensor 2.
Parameters:

a. nCattemp22(OUT)
Type : UCHAR8
Extracted Catalyst Temperature Bank 2, Sensor 2 value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

22. GET_VPWR(flVpwr,puchBuf)

This macro is used to extract the Control Module Voltage.

Parameters:

a. flVpwr(OUT)
   Type : FLOAT32
   Extracted the Control Module Voltage value is given in this parameter.

b. puchBuf(IN):
   Type : UCHAR8[]
   Buffer containing the OBD data.

23. GET_LOAD_ABS(nLoadAbs,puchBuf)

This macro is used to extract the Absolute Load Value.

Parameters:

a. nLoadAbs(OUT)
   Type : UCHAR8
   Extracted the Absolute Load Value is given in this parameter.

b. puchBuf(IN):
   Type : UCHAR8[]
   Buffer containing the OBD data.

24. GET_LAMBDA(flLambda,puchBuf)

This macro is used to extract the AFuel/Air commanded Equivalence Ratio.

Parameters:

a. flLambda(OUT)
   Type : UCHAR8
   Extracted the Fuel/Air commanded Equivalence Ratio is given in this parameter.

b. puchBuf(IN):
   Type : UCHAR8[]
   Buffer containing the OBD data.

25. GET_TP_R(uchTpR,puchBuf)

This macro is used to extract the Relative Throttle Position.

Parameters:

a. uchTpR,(OUT)
Type : FLOAT32
Extracted the Relative Throttle Position value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

26. GET_AAT(nAat,puchBuf)

This macro is used to extract the Ambient Air Temperature .
Parameters:

a. nAat(OUT)
Type : UCHAR8
Extracted the Ambient Air Temperature value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

27. GET_TP_B(uchTpB,puchBuf)

This macro is used to extract the Absolute Throttle Position B.
Parameters:

a. uchTpB(OUT)
Type : UCHAR8
Extracted theAbsolute Throttle Position B value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

28. GET_TP_C(uchTpC,puchBuf)

This macro is used to extract the Absolute Throttle Postion C.
Parameters:

a. uchTpC(OUT)
Type : UCHAR8
Extracted theAbsolute Throttle Position C value is given in this parameter.

b. puchBuf(IN):

Type : UCHAR8[]
Buffer containing the OBD data.

29. GET_APP_D(uchAppD,puchBuf)

This macro is used to extract the Accelerator Pedal Position D.
Parameters:

a. uchAppD(OUT)
Type : UCHAR8
Extracted theAccelerator Pedal Position D value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

30. GET_APP_E(uchAppE,puchBuf)

This macro is used to extract the Accelerator Pedal Position D.
Parameters:

a. uchAppD(OUT)
Type : UCHAR8
Extracted theAccelerator Pedal Position D value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

31. GET_APP_F(uchAppF,puchBuf)

This macro is used to extract the Accelerator Pedal Position F .
Parameters:

a. uchAppF(OUT)
Type : UCHAR8
Extracted the Accelerator Pedal Position F value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

32. GET_TAC_PCT(uchTacPct,puchBuf)

This macro is used to extract the Commanded Throttle Actuator Control .
Parameters:

a. uchAppF(OUT)
Type : UCHAR8
Extracted the Accelerator Pedal Position F value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

33. GET_MIL_TIME(unMilTime,puchBuf)

This macro is used to extract the Engine run time while MIL is activated .
Parameters:

a. unMilTime(OUT)
Type : UCHAR8
Extracted the Engine run time value while MIL is activated is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

34. GET_CLR_TIME(unClrTime,puchBuf)

This macro is used to extract the Engine run time since DTCs cleared.
Parameters:

a. unMilTime(OUT)
Type : UCHAR8
Extracted Engine run time value since DTCs cleared is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

35. GET_ALCH_PCT(uchAlchPct,puchBuf)

This macro is used to extract the Alcohol Fuel Percentage rate.

Parameters:

a. uchAlchPct(OUT)
   Type : UCHAR8
   Extracted Alcohol Fuel Percentage value is given in this parameter.

b. puchBuf(IN):
   Type : UCHAR8[]
   Buffer containing the OBD data.

36. GET_APP_R(uchAppR,puchBuf)

This macro is used to extract the Relative accelerator pedal position .

Parameters:

a. uchAppR(OUT)
   Type : UCHAR8
   Extracted Relative accelerator pedal position value is given in this parameter.

b. puchBuf(IN):
   Type : UCHAR8[]
   Buffer containing the OBD data.

37. GET_BAT_PWR(uchBatPwr,puchBuf)

This macro is used to extract the Hybrid Battery Pack Remaining Life .

Parameters:

a. uchBatPwr(OUT)
   Type : UCHAR8
   Extracted Hybrid Battery Pack Remaining Life value is given in this parameter.

b. puchBuf(IN):
   Type : UCHAR8[]
   Buffer containing the OBD data.

38. GET_EOT(nEot,puchBuf)

This macro is used to extract the Engine Oil Temperature value .

Parameters:

a. nEot(OUT)
   Type : UCHAR8

Extracted the Engine Oil Temperature value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

39. GET_FUEL_TIMING(flFuelTiming,puchBuf)

This macro is used to extract the Fuel Injection Timing value .
Parameters:

a. flFuelTiming(OUT)
Type : UCHAR8
Extracted the Fuel Injection Timing value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

40. GET_FUEL_RATE(flFuelRate,puchBuf)

This macro is used to extract the Engine Fuel Rate .
Parameters:

a. flFuelRate(OUT)
Type : UCHAR8
Extracted the Engine Fuel Rate value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

41. GET_TQ_DD(chTqDD,puchBuf)

This macro is used to extract the Driver's Demand Engine Percent Torque rate .
Parameters:

a. chTqDD(OUT)
Type : UCHAR8
Extracted the Driver's Demand Engine Percent Torque Rate value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]

Buffer containing the OBD data.

42. GET_TQ_ACT(chTqAct,puchBuf)

This macro is used to extract the Actual Engine Percent Torque .
Parameters:

a. chTqAct(OUT)
Type : UCHAR8
Extracted the Actual Engine Percent Torque value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

43. GET_TQ_REF(unTqRef,puchBuf)

This macro is used to extract the Engine Reference Torque .

Parameters:

a. unTqRef(OUT)
Type : UCHAR8
Extracted the Engine Reference Torque value is given in this parameter.

b. puchBuf(IN):
Type : UCHAR8[]
Buffer containing the OBD data.

DTS-OBDStack Driver interface

CAN Driver Interface of the stack is specified in the CANLib.h file. Driver interface consists of few functions which need to be implemented by the CAN driver for the target board. Driver should be capable of operating at 250 Kbps and 500 Kbps and should support 11 bit and 29 bit ID. It should also support setting minimum one filter and mask values.

The functions are explained below

1. INT16 CAN_Init(ULONG32 Baud,UCHAR8 ID_Type,ULONG32 Filter_ID,ULONG32 Mask);

Stack uses this function to initialize the CAN driver.

Parameters:

a. Baud (IN)
    Type : ULONG32
    Baud rate to be used by the CAN driver.

b. ID_Type (IN):
    Type : UCHAR8
    This can be CAN_11BIT or CAN_29BIT

c. Filter_ID (IN):
    Type : ULONG32
    Filter value to be used

d. Mask(IN):
    Type : ULONG32
    Mask value to be used

2. INT16 CAN_DeInit(void);

Stack uses this function to deinitialize the driver.

3. INT16 CheckForCANErrors(void);

Stack uses this function to check for any CAN errors. Driver should return the error code if some error is present

4. INT16 CANReceive(CANMsg* pMsg);

Stack uses this function receive a CANMessage.

Parameters:

a. pMsg (OUT)
    Type : CANMsg*
    Pointer to the CANMsg Buffer. Driver should copy the received message in this buffer

Function should return the number of CAN Messages available for reading and should return 0 if there are no messages to read.

5. INT16 CANTransmit(CANMsg* pMsg);

Stack uses this function transmit a CANMessage.

Parameters:

a. pMsg (IN)

Type : CANMsg*
Pointer to the CANMsg Buffer for transmission

Usage Examples

Stack initialization

Call InitOBD() function to initialize the stack. This function initializes the CAN driver and internal stack variables. It determines the protocols to be used and collects information regarding the supported parameters.

Reading Mode 1 parameters

Mode 1 is used to get engine parmaters. Following approach should be used to read Mode-1parameters.

1. Call OBDSendRequest(INT16 nMode, UCHAR8 uchParam);

   nMode should be   MODE_ENGINE_PARAM and uchParam should be the required parameter. All the mode 1paramters are defined in the OBDStack.h file

2. Call OBDReadReply(UCHAR8** ppBuf, INT16* pnLen) function periodically until it returns Success of failure.
   It returns the data buffer and data length.

3. Macros are provided to extract many of the parameters. If a Macro is available pass the buffer to the macro to extract the value. If not, the value has to be extracted programmatically.

Reading DTC information

1. Call OBDSendRequest(INT16 nMode, UCHAR8 uchParam);

   nMode should be   MODE_GET_DTC and uchParam is ignored in the stack and is suggested to set 0xFF

2. Call OBDReadReply(UCHAR8** ppBuf, INT16* pnLen) function periodically until it returns Success of failure.
   It returns the data buffer and data length.

3. Call ParseDTC(UCHAR8 DtcBuf[][5], UCHAR8* puchBuf) passing the buffer pointer obtained from step 2.

   This function returns the number of DTCs as the return value and then provides DTC codes in the 2 dimensional array.

Reading VIN Number

1. Call OBDSendRequest(INT16 nMode, UCHAR8 uchParam);

   nMode should be   MODE_VEHICLE_INFO and uchParam is ignored in the stack and is suggested to set 0xFF

2. Call OBDReadReply(UCHAR8** ppBuf, INT16* pnLen) function periodically until it returns Success of failure.
   It returns the data buffer and data length.

3. Call GetVIN(UCHAR8* VinBuf, UCHAR8* puchBuf) passing the buffer pointer obtained from step 2.

   This function returns the VIN number in the VinBuf parameter.

DTS-OBDStack Internal Functions

OBD Application layer

OBD Layer API is used by the application program

The functions used in CanOBDM1 file are listed below

INT16 CANQueryParamSet()

INT16 CANInitOBD()

INT16 CANReadReply(UINT16* punLen)

INT16 CANSendRequest(UCHAR8 uchMode, UCHAR8 uchParam)

INT16 CheckNLReply()

BOOL IsSupported(UCHAR8 uchParam)

void SendOBDRequest()

INT16 CANGetParamSupport(UCHAR8 uchPID)

OBD Layer APPLICATION LAYER Function Reference

- INT16 CANInitOBD()

  This Function is used to Initialize the CAN driver
  Check the supported protocols in the following order
  * 11 bit ID – 500Kbps
  * 29bit ID- 500 Kbps
  * 11bit-250 kbps
  * 29 bit-250 kbps

  parameters: Nil
  Return Value: Function returns CAN_OBD_INIT_FAIL if failed or
  CAN_OBD_INIT_SUCCESS if success
  Data type : INT16

- INT16 CANQueryParamSet()

  This Function is used to Builds the supported parameter list by querying the ECU

  parameters: Nil
  Return Value: Function returns CAN_OBD_INIT_FAIL if failed or
  CAN_OBD_INIT_SUCCESS if success
  Data type : INT16

- INT16 CANGetParamSupport(UCHAR8 uchPID)

  This Function is used to Builds the supported parameter list by querying the ECU

  parameters: Nil
  Return Value: Function returns CAN_OBD_INIT_FAIL if failed or
  CAN_OBD_INIT_SUCCESS if success
  Data type : INT16

- INT16 CANGetParamSupport(UCHAR8 uchPID)

This Function is used to Gets the supported parameter for the given PID.
This function blocks till the reply is obtained or time out happens

Parameters : uchPID(IN)
Data type : UCHAR8 PID values (0x00,0x20, 0x40,etc)
Return values : one of the receive error codes if failed or
PARAM_REQ_SUCCESS if success
Data type : INT16

- INT16 CANSendRequest(UCHAR8 uchMode, UCHAR8 uchParam)

  This Function is used to Stores the Mode and PID to global variables and for Mode 1 checks whether the parameter is supported. If supported calls SendOBDRequest function to send the OBD request to ECU

  Parameters :
  uchMode : OBD Mode(IN)
  Data type: UCHAR8
  uchParam: OBD PID (IN)
  Data type: UCHAR8
  Returns : OBD_PARAM_NOT_SUPPORTED if parameter is not supported
  OBD_REQ_SUCCESS if success
  Data Type : INT16

- INT16 CheckNLReply()

  This Function is used to checks whether the reply from obd stack success or not
  Parameters : None
  Returns : OBD_RCV_ERROR code if failed
  OBD_RCV_SUCCESS if success
  Data Type : INT16

- void SendOBDRequest()

  This Function is used to configure the CAN Module
  Set 11 bit or 29 bit ID
  Parameters : None
  Returns : None

- BOOL IsSupported(UCHAR8 uchParam)

  This Function Stores the Mode and PID to global variables and for Mode 1 checks whether the parameter is supported.
  Parameters :
  uchParam: OBD PID (IN)
  Data Type: UCHAR8
  Returns : BOOL type, if TRUE for supported parameters
  and false for non supported

The functions used in CanOBDN1 file are listed below

- INT16 SendOBDNlFrame(unsigned char* pBuf,UINT16 nSize,ULONG32 lCANID);
- INT16 ReceiveOBDNlFrame(UCHAR8*pBuf, UINT16* pnSize);
- void SendFCframe(void);

OBD Layer NETWORK LAYER Function Reference

This layer implements the ISO15765-2 network layer specifications

INT16 SendOBDNlFrame(unsigned char* pBuf,UINT16 nSize,ULONG32 lCANID)

This function is used to send CAN REquest data to the Network frame and check size and determine single frame or multi frame.

Parameters:
pBuf (IN) :
Data type : **unsigned char**

OBD message is stored in this parameter.

nSize (IN):
Data type : **UINT16**
Data Length code

lCANID(IN):

Data type : **ULONG32**

CAN Request ID I stored in this parameter

Return Value:

Data type : INT16 type

if CAN data transmit success nRet returns success

and data transmission failed it returns FALSE

INT16 ReceiveOBDNlFrame(UCHAR8*pBuf, UINT16* pnSize)

This function is used to receive data from the network layer and check size and determine single frame or multi frame .

Parameters:

pBuf(OUT) :

Data type : **UCHAR8**

OBD message is stored in this parameter.

pnSize(OUT):

Data type : **UINT16**

store the size of data.

Return Value:

Data type : INT16 type

if CAN data receive success nRet returns NL_RCV_SUCCESS

and data transmission failed it returns NL_NOMSG

void SendFCframe(void)

This function is used to Transmit the CAN message and send it to the network .

Parameters: Nill

Return Value: Nill

Stack configuration

These are the Header file for the OBD Stack configuration. It is Editable to the stack user. The Macros that used in this file are listed given below

OBD_BUF_SIZE - Obd buffer is used to store the incoming data from the CAN network. Here we assighn the size is 100 bytes. Size should be tuned for the application requirements.

CANID_11BIT - For 11 bit Standard ID can message. Usually the value for SID PID is 0x7E0

CANID_29BIT - For 29 bit Extended ID can message. Usually the value for EID PID is 0x18DB33F1

OBD_TIMEOUT - Defines 100 msec time out per request

OBD_MF_TIMEOUT - Defines 3 sec time out to receive an OBD multi frame message

SYSTEM_FREQ – Defines 48Mhz crystal oscillator frequency.

DataTypedefenitions

Header file defining Datatypes. Will change as per the architecture
STACK SUPPORTED DATATYPES AND SIZES ARE GIEVEN BELOW

CHAR8 - 8bit signed char
UCHAR8 - 8bit unsigned char
INT16 - 16 bit signed
UINT16 - 16 bit ubnsigned int
LONG32 - 32 bit signed
ULONG32 - 32 bit unsigned int
FLOAT32 - 32 bit float type

Porting and using DTS-OBD stack

Porting of DTS-OBD stack is a simple process and the steps are given below.

1. Write the CAN driver for the target board as per the driver interface explained
2. Modify the DataTypeDef.h file to use the data types for the target processor
3. Edit the StackConfig.h file to define appropriate buffer sizes
4. Organize the files in the Platform folder as given in the file organization section
5. Write the OBD application following the structure given in the sample application code