API Document for OBDII Library

13th-April-2022

AUTHOR(S)

Name	Function	Organisation	Date	Signature
Sahana	Member Technical	iWave Systems Technologies Pvt. Ltd.	13-04-2022	

APPROVAL

Name	Function	Organisation	Date	Signature
Rizwan	Project Lead	iWave Systems Technologies Pvt. Ltd.	13-04-2022	

DISTRIBUTION	iWave Systems Technologies Pvt. Ltd.

CONTACT INFO

Name	Telephone	E-mail
iWave Systems Technologies Pvt Ltd, 7/B, 29th Main, BTM Layout, 2nd Stage, Bangalore –560 076, India.	+91-80-2668-3700 +91-80-2678-1643	

DOCUMENT IDENTIFICATION

Project Name	PRGET
Document Name	iW-PRGET-UM-01-R3.0-REL2.8_OBDII_Library_API_Reference
Document Home	
Status	Version 2.8

DOCUMENT REVISION HISTORY

Revision	Date	Description	Author	Reviewer(s)
Rev3.0/Rel2.1	**26-06-21**	**API Reference User Manual**	**Nitin**	**Sheik Ajith**
Rev3.0/Rel2.2	**21-10-21**	**API Reference User Manual**	**Nitin**	**Sheik Ajith**
Rev3.0/Rel2.3	**20-01-22**	**API Reference User Manual**	**Jesmitha**	**Anjali**
Rev3.0/Rel2.4	**11-02-22**	**API Reference User Manual Updated with details of API calling sequence.**	**Anushree**	**Anjali**
Rev3.0/Rel2.5	**15-02-22**	**API Reference User Manual updated with battery API return value and init (1) constraints.**	**Anushree**	**Anjali**
Rev3.0/Rel2.6	**09-03-22**	**API Reference User Manual Updated with** • **Hexadecimal conversion in case of python** • **Details about init (1)** • **can_read (), can_write () description and CANFD insertion steps.** • **Format changes of get_gsm_sim_status ()** • **Return codes related to sim**	**Anushree**	**Anjali**
Rev3.0/Rel2.7	**30-03-22**	**API Reference User Manual updated with** • **API Specific return codes.** • **APIs related to magnetometer** • **Example of calling each API.**	**Anushree**	**Anjali**
Rev3.0/Rel2.8	**13-04-22**	• **Format Changes** • **Example for set_acc_low_pass_filter and set_acc_sampling_frequency** • **Added note for Agps_init()**	**Sahana**	**Anjali**

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose and Scope

The purpose of this document is to describe the API's supported by the OBDII library for developing an application. Developers can use this document while developing application using OBDII library.

1.2 Glossary

Acronyms	Description
API	Application Programming interface
CAN	Controller Area Network
IPC	Inter Process Communication

Table 1: Acronyms & Abbreviations

2 Functional Description

This chapter describes APIs supported by the OBDII library.

NOTE:

- All the below API's return code are with respect to C language. Print the return values in hexadecimal.
- If python is used, then 2's compliment of hexadecimal value has to be done to get the proper return value.

  Example:

  ret= init ()

  print (hex((ret + (1 << 32)) % (1 << 32))

2.1 General APIs

2.1.1 init

Init	
Description	This function initializes the IPC mechanism required by OBDII library.
Syntax	int init (int network_enable)
Arguments	network_enable – Automatic network monitor feature will be enabled/disabled based on the value passed on to this variable. ▪ If network_enable = 0: Automatic network monitor feature would be disabled. ▪ If network_enable = 1: Automatic network monitor feature will be enabled.
Return	Refer General Return Codes section.
Example	ret=init (0);
NOTE	• **If "0" is passed as argument:** ▪ 4G module initialization and connection establishment would have to handled manually using the provided APIs. (Refer Sequence to be followed if 0 is passed to init () for calling sequence). • **If "1" is passed as argument:** ▪ 4G module initialization and connection establishment will be internally without any external intervention. ▪ Please note that when this feature is enabled, "network_monitor_disable ()" would

	have to be called before calling the main deinitialization function (deinit ()). ▪ This can be used only if it is needed since there are few constraints like some API cannot be used since it will be used internally by network manager thread. ▪ Below are the APIs which has limitations with init (1): o establish_connection o set_gsm_flight_mode_on o set_gsm_flight_mode_off o gsm_modem_on o gsm_modem_off o set_gsm_network_mode

2.1.2 get_time

get_time	
Description	This API reads the date & time from the OBDII.
Syntax	void get_time (char *buf)
Arguments	▪ char * buf – pointer to character array to store date & time of the OBD-II device.
Return	None
Example	char buf [50] = {0}; ret = get_time(&buf);
NOTE	-

2.1.3 check_adc_voltage

check_adc_voltage	
Description	This API reads the adc external supply voltage.
Syntax	int check_adc_voltage (double *volt)
Arguments	▪ double * volt – Voltage value.
Return	Refer General Return Codes section.
Example	double volt; ret = check_adc_voltage(&volt);
NOTE	-

2.1.4 ntp_server_update

ntp_server_update	
Description	This API updates the device time to standard UTC time using ntp time server.
Syntax	int ntp_server_update ()
Arguments	None
Return	Refer General Return Codes section.
Example	ret = ntp_server_update ();
NOTE	▪ Make sure that device is having internet connection before calling this API. ▪ The NTP service will be enabled by default on boot up. ▪ If the "ntp_server_update" API has to be used in the application please make sure to disable the NTP service by executing the below commands: **"systemctl disable ntpd"** **"systemctl daemon-reload"** **"sync"** ▪ Reboot the device. ▪ After device reboots, NTP service will be disabled and "ntp_server_update" API can be used.

2.1.5 led_enable

led_enable	
Description	This API is used to turn on the green LED.
Syntax	int led_enable ();
Arguments	None
Return	Refer General Return Codes section.
Example	ret = led_enable ();
NOTE	-

2.1.6 led_disable

led_disable	
Description	This API is used to turn off the green LED.
Syntax	int led_disable ();
Arguments	None
Return	Refer General Return Codes section.
Example	ret = led_disable ();
NOTE	-

2.1.7 restart_device

restart_device	
Description	This API is used to restart the OBDII device.
Syntax	int restart_device ();
Arguments	None
Return	Refer General Return Codes section.
Example	ret = restart_device ();
NOTE	-

2.1.8 i2c_write

i2c_write	
Description	This API is used to write values to i2c registers.
Syntax	int i2c_write (int bus_num, uint8_t slave_addr, uint8_t data_addr, uint8_t value)
Arguments	int bus_num – i2c bus number uint8_t slave_addr – slave address uint8_t data_addr – data address of register uint8_t value – value to be written to the data address
Return	Refer General Return Codes section.
Example	#define SLAVE_ADDR 0x6a #define DATA_ADDR 0x20 #define VALUE 0x86

	Main () {
	int bus_num =1;
	ret = i2c_write (bus_num, SLAVE_ADDR, DATA_ADDR, VALUE);
	}

NOTE	▪ The bus number should be valid. Refer "I2C Bus Number For Devices" Section. ▪ The addresses passed to the API should be in hex format and should be valid.

2.1.9 i2c_read

i2c_read	
Description	This API is used to read data from i2c registers.
Syntax	int i2c_read (int bus_num, uint8_t slave_addr, uint8_t data_addr, uint8_t *value)
Arguments	int bus_num – i2c bus number Uint8_t slave_addr – slave address uint8_t data_addr – data address of register uint8_t *value – this variable gives the value in the data address register
Return	Refer General Return Codes section.

Example	#define SLAVE_ADDR 0x6a
	#define DATA_ADDR 0x20
	Main () {
	int bus_num = 1;
	uint8_t resp;
	ret = i2c_read (bus_num, SLAVE_ADDR, DATA_ADDR, &resp);
	}

NOTE	▪ The bus number should be valid. Refer "I2C Bus Number For Devices" Section. ▪ The addresses passed to the API should be in hex format and should be valid.

2.1.10 get_mac_address

get_mac_address	
Description	This API used to get the MAC address of interfaces.
Syntax	int get_mac_address (char *interface, char *mac_address)
Arguments	char *interface – interface name. Eg: “wlan0”, “eth0”, “hci0”, etc.
Return	Refer General Return Codes section.
Example	char wifi_addr [30]; ret = get_mac_adddress (“wlan0”, wifi_addr);
NOTE	-

2.1.11 get_cpu_id

get_cpu_id	
Description	This API used to get the CPU ID.
Syntax	int get_cpu_id (char *cpuid, int cpuid_len)
Arguments	▪ char *cpuid – array to store the CPU id. ▪ int cpuid_len – Size/Length of the array to store the CPU ID.
Return	Refer General Return Codes section.

Example	char cpu_id [50] = {0};
	int cpuid_length = 40;
	ret = get_cpu_id (cpu_id, cpuid_length);

NOTE	Array size cannot be less than 23.

2.1.12 config_timer_wakeup

config_timer_wakeup	
Description	This API is used to enable or disable timer wakeup.
Syntax	int config_timer_wakeup (int option, int timer)
Arguments	▪ int option – option to enable or disable timer wakeup 1 – enable 0 – disable

	▪ int timer – timer value in seconds Ex: 10
Return	Refer General Return Codes section.
Example	int option = 1; ret = config_timer_wakeup (option, 60);
NOTE	-

2.1.13 push_device_to_sleep

push_device_to_sleep	
Description	This API is used to put device to sleep mode.
Syntax	int push_device_to_sleep ()
Arguments	None
Return	Refer General Return Codes section.
Example	ret = push_device_to_sleep ();
NOTE	▪ Any wakeup API has to be called before calling this API. ▪ If none of the wakeup APIs are called the device need to be power cycled manually.

2.1.14 ign_pin_status_check_enable

ign_pin_status_check_enable	
Description	This API is used to enable the ignition event status thread.
Syntax	int ign_pin_status_check_enable ()
Arguments	None
Return	Refer General Return Codes section.
Example	ret = ign_pin_status_check_enable ();
NOTE	-

2.1.15 ignition_pin_status

ignition_pin_status	
Description	This API is used to check the ignition pin status.
Syntax	int ignition pin_status ()
Arguments	None
Return	Refer General Return Codes section.
Example	ret = ignition_pin_status ();
NOTE	-

2.1.16 ign_pin_status_check_disable

ign_pin_status_check_disable	
Description	This API is used to disable the ignition event status thread.
Syntax	int ign_pin_status_check_disable ()
Arguments	None
Return	Refer General Return Codes section.
Example	ret = ign_pin_status_check_disable ();
NOTE	▪ ign_pin_status_check_enable () should be called before ignition_pin_status (). **Following are the return code expected:** ▪ ign_pin_status returns 1 in the presence of ignition. ▪ ign_pin_status returns -1 in the absence of ignition. ▪ If ign_pin_status_check_enable () is not called before ignition_pin_status (), status value is always 0. (Even when ignition status is toggled). ▪ If ign_pin_status_check_disable () is called before ign_pin_status (), status value is always 0.

2.1.17 config_ignition_wakeup

config_ignition_wakeup	
Description	This API is used to enable or disable ignition wakeup.
Syntax	int config_ignition_wakeup (int option)

Arguments	▪ int option – option to enable or disable ignition wakeup 1 – enable 0 – disable
Return	Refer General Return Codes section.
Example	int option = 1; ret = config_ignition_wakeup (option);
NOTE	-

2.1.18 deinit

Deinit	
Description	Please note that when this feature is enabled, “network_monitor_disable ()” should be called before calling the main deinitialization function (deinit ()). This function de-initializes the IPC mechanism required by OBDII library.
Syntax	int deinit ()
Arguments	None
Return	Refer General Return Codes section.
Example	ret = deinit ();
NOTE	▪ This API will deinitialize Ethernet and turn off LED.

2.2 Cellular module APIs

2.2.1 gsm_modem_on

gsm_modem_on	
Description	This API is used to turn on the GSM modem.
Syntax	int gsm_modem_on (char *cpin, int length)
Arguments	▪ char *cpin – CPIN value of the SIM that is being used. If the SIM does not have CPIN value, “0000” can be passed to the same argument. ▪ int length – Size/Length of the CPIN value that is being passed.
Return	Refer Cellular module specific return codes section.
Example	ret = gsm_modem_on (“0000”, 4);
NOTE	-

2.2.2 gsm_modem_off

gsm_modem_off	
Description	This API is used to turn off the GSM modem.
Syntax	int gsm_modem_off ()
Arguments	None
Return	Refer Cellular module specific return codes section.
Example	ret = gsm_modem_off ();
NOTE	-

2.2.3 get_gsm_imei

get_gsm_imei	
Description	This API is used to get the IMEI number of the GSM module.
Syntax	int get_gsm_imei (char *imei_no, int length)
Arguments	▪ char *imei_no - array to store the IMEI number. ▪ int length – Size/Length of the array to store the IMEI number.
Return	Refer Cellular module specific return codes section.
Example	char imei [20] = {0}; ret = get_gsm_imei (imei, sizeof (imei));
NOTE	▪ Array size cannot be less than 16. ▪ gsm_modem_on () should be called before calling this API.

2.2.4 check_gsm_nw_connection

check_gsm_nw_connection	
Description	This function is used to check whether is internet connectivity is active or not.
Syntax	int check_gsm_nw_connection ()
Arguments	None
Return	Refer Cellular module specific return codes section.
Example	ret = check_gsm_nw_connection ();
NOTE	▪ gsm_modem_on () should be called before calling this API.

2.2.5 check_gsm_modem_status

check_gsm_modem_status	
Description	This API is used to check whether modem is on or off.
Syntax	int check_gsm_modem_status ()
Arguments	None
Return	0 if modem is ON Refer Cellular module specific return codes section.
Example	ret = check_gsm_modem_status ();
NOTE	▪ gsm_modem_on () should be called before calling this API.

2.2.6 set_gsm_flight_mode_on

set_gsm_flight_mode_on	
Description	This API is used to set the flight mode ON.
Syntax	int set_gsm_flight_mode_on ()
Arguments	None
Return	Refer Cellular module specific return codes section.
Example	ret = set_gsm_flight_mode_on ();
NOTE	▪ gsm_modem_on () should be called before calling this API.

2.2.7 set_gsm_flight_mode_off

set_gsm_flight_mode_off	
Description	This API is used to set the flight mode OFF.
Syntax	int set_gsm_flight_mode_off ()
Arguments	None
Return	Refer Cellular module specific return codes section.
Example	ret = set_gsm_flight_mode_off ();
NOTE	▪ gsm_modem_on () should be called before calling this API.

2.2.8 establish_connection

establish_connection	
Description	This API establishes ppp network connection.
Syntax	int establish_connection ()
Arguments	None
Return	Refer Cellular module specific return codes section.
Example	ret = establish_connection ();
NOTE	▪ gsm_modem_on () should be called before calling this API.

2.2.9 get_gsm_sim_status

get_gsm_sim_status	
Description	This API will check whether the sim is inserted in device or not.
Syntax	int get_gsm_sim_status (int *sim_status_val)
Arguments	▪ sim_status_val- integer pointer which stores the current sim status.
Return	Refer Cellular module specific return codes section.
Example	int sim_status_val; ret = get_gsm_sim_status (&sim_status_val);
NOTE	▪ gsm_modem_on () should be called before calling this API.

2.2.10 get_gsm_sim_iccid

get_gsm_sim_iccid	
Description	This API to get the ICCID of the GSM module.
Syntax	int get_gsm_sim_iccid (char *iccid, int length)
Arguments	▪ char *iccid – array to store the ICCID of the GSM module. ▪ int length – Size/Length of the array to store the ICCID of the GSM module.
Return	Refer Cellular module specific return codes section.
Example	char iccid [30] = “”; ret = get_gsm_sim_iccid (iccid, sizeof(iccid);
NOTE	▪ Array size cannot be less than 25. ▪ gsm_modem_on () should be called before calling this API.

2.2.11 set_gsm_network_mode

set_gsm_network_mode	
Description	This API is used to change the network mode.
Syntax	int set_gsm_network_mode (int type)
Arguments	▪ int type – contains the network mode to be set.
Return	Refer Cellular module specific return codes section.
Example	int option = 1; ret = set_gsm_network_mode (int option)
NOTE	▪ gsm_modem_on () and establish_connection () should be called before calling this API. Auto = 1 2G = 2 3G = 3 4G = 4

2.2.12 get_gsm_signal_strength

get_gsm_signal_strength	
Description	This API to get the signal range details of the network.
Syntax	int get_gsm_signal_strength (char *signal_lvl, int length)
Arguments	▪ char *signal_lvl – array to store the signal range. ▪ int length – Size/Length of the array to store the signal range.
Return	Refer Cellular module specific return codes section.
Example	char arr_sig_strength [20] = {0}; ret = get_gsm_signal_strength (arr_sig_strength, sizeof(arr_sig_strength));
NOTE	▪ gsm_modem_on () should be called before calling this API. ▪ Make sure that sim is inserted before calling this API.

2.2.13 get_gsm_nw_reg

get_gsm_nw_reg	
Description	This API to get the SIM registration details.
Syntax	int get_gsm_nw_reg (char * cell_id, int cell_id_len, char *lac, int lac_len)
Arguments	▪ char *cell_id – array to store the cell identification. ▪ int cell_id_len – Size/Length of the array to store the cell identification ▪ char *lac – array to store location area code. ▪ int lac_len – Size/Length of the array to store location area code
Return	Refer Cellular module specific return codes section.
Example	char cell_id [20] = {0}; char lac [20] = {0}; ret = get_gsm_nw_reg (cell_id, sizeof(cell_id), lac, sizeof(lac));
NOTE	▪ gsm_modem_on () should be called before calling this API. ▪ Make sure that sim is inserted before calling this API. ▪ Array size cannot be less than 20.

2.2.14 gsm_at_cmd

gsm_at_cmd	
Description	This API is used to send AT commands and receive the response.
Syntax	int gsm_at_cmd (char *at_cmd, char *resp, int length, int max_resp_time)
Arguments	▪ char *at_cmd – AT command to be sent. ▪ char *resp – Array to store the AT command response. ▪ int length – Size/Length of array to store AT command response. ▪ int max_resp_time – Response wait time in milliseconds (default max_resp_time can be considered as 300ms (300000), if it is not mentioned in AT command reference manual).
Return	Refer Cellular module specific return codes section.
Example	char at_cmd [] = “at+pin?”; char resp [200]; int length;

	ret = gsm_at_cmd (at_cmd, &resp, length, 500000); //500000 is 500ms
NOTE	▪ gsm_modem_on () should be called before calling this API. ▪ Example for AT commands: AT+CPIN? , AT+CREG?

2.2.15 gsm_apn_configuration

gsm_apn_configuration	
Description	This API is used to change the APN configurations of the SIM.
Syntax	void gsm_apn_configuration (char *apn_name, char *atd_num, char *username, char *password)
Arguments	▪ char *apn_name – APN name to be set. Eg: “airtelgprs.com” ▪ char *atd_num – ATD number to be set. Eg: “ATDT*99***1#” ▪ char *username – Username of the SIM. ▪ char *password – Password of SIM.
Return	None
Example	char apn [] = “airtelgprs.com” char atd_num [] = “ATDT*99***1#” char username [] = “xxxxx” char password [] = “yyyyyy” ret = gsm_apn_configuration (&apn, &atd_num, &username, &password);
NOTE	▪ gsm_modem_on () should be called before calling this API. ▪ Pass NULL if there is no Username and Password for sim.

2.2.16 check_network_connection

check_network_connection	
Description	This API is used to check the network availability.
Syntax	int check_network_connection ()
Arguments	None
Return	Refer Cellular module specific return codes section.
Example	ret = check_network_connection ();
NOTE	-

2.2.17 GSM_set_to_message_init

GSM_set_to_message_init	
Description	This API is used to set the GSM to text message mode.
Syntax	int GSM_set_to_message_init ()
Arguments	None
Return	Refer Cellular module specific return codes section.
Example	ret = GSM_set_to_message_init ();
NOTE	-

2.2.18 network_monitor_disable

network_monitor_disable	
Description	This API is used to disable the Automatic 4G monitor feature.
Syntax	int network_monitor_disable ()
Arguments	None
Return	Refer Cellular module specific return codes section.
Example	ret = network_monitor_disable ();
NOTE	▪ This API would have to be called before calling the deinit ().

2.2.19 unread_message

unread_message	
Description	This API is used to read all unread SMS.
Syntax	int unread_message (char *msg_buf, int length, int max_resp_time)
Arguments	▪ char *msg_buf – array to store all unread messages. ▪ Length – Size/Length of the array to store all unread messages ▪ max_resp_time – response time to read response from GSM module. (In millisec)
Return	Refer Cellular module specific return codes section.
Example	char resp_buffer [1024]; ret = unread_message (resp_buffer, sizeof(resp_buffer), 800000);
NOTE	-

2.2.20 read_message

read_message	
Description	This API is used to read all read SMS.
Syntax	int read_message (char *msg_buf, int length, int max_resp_time)
Arguments	▪ char *msg_buf – array to store all read messages. ▪ Length – Size/Length of the array to store all read messages ▪ max_resp_time – response time to read response from GSM module. (In microseconds)
Return	Refer Cellular module specific return codes section.
Example	char resp_buffer [1024]; ret = read_message (resp_buffer, sizeof(resp_buffer), 800000);
NOTE	-

2.2.21 send_sms

send_sms	
Description	This API is used to send SMS to specified mobile number.
Syntax	int send_sms (char *msg_response, char *sender_number, int max_resp_time)
Arguments	▪ char *msg_response – array stored with message to send. ▪ char *sender_number – array stored with mobile number in which message has to be to send. ▪ Max_resp_time – response time to read response from GSM module. (In microseconds)
Return	Refer Cellular module specific return codes section.
Example	ret = send_sms (“Hi, I have a user application”, “7089234567”, “800000);
NOTE	▪ gsm_modem_on () and establish_connection () should be called before calling send_sms ().

2.2.21 delete_message

delete_message	
Description	This API is used to delete SMS using index number.
Syntax	int delete_message (int index, int max_resp_time)
Arguments	▪ int index – index number of the message to be deleted. ▪ Max_resp_time – response time to read response from GSM module. (In microseconds)
Return	Refer Cellular module specific return codes section.
Example	ret = delete_message (1, 800000);
NOTE	-

2.2.22 delete_all_messages

delete_all_messages	
Description	This API is used to delete all read SMS.
Syntax	int delete_all_messages (int max_resp_time)
Arguments	▪ max_resp_time – response time to read response from GSM module. (In microseconds)
Return	Refer Cellular module specific return codes section.
Example	ret = delete_all_messages (1, 800000);
NOTE	-

2.3 Cellular module specific return codes

Return Codes	Description
E0010001	GSM port initialization error
E0010002	GSM port write error
E0010003	GSM port read error
E0010004	GSM port disconnection error
E0010005	GSM AT port initialization error
E0010006	GSM AT port write error
E0010007	GSM AT port read error

E0010008	IMEI read timeout error
E0010009	Network connection down
E001000A	SIM is not detected
E001000B	SIM detected
E001000C	SIM status unknown
E1010001	GSM SIM registration error
E1010002	Failed to obtain signal strength
E1010003	Failed to obtain SIM ICCID
E1010004	GSM SMS send error
E1010005	GSM SMS send buffer invalid
E1010006	GSM SMS delete using index error
E1010007	GSM SMS delete all read messages error
E1010008	Data Overflow occurred – Length of buffer passed insufficient
E1010009	GSM Network mode enabled (Network related handling done internally)
E2010001	GSM semaphore initialization error
E2010002	SIM is not valid
E2010003	GSM NTP Fail
E2010004	GSM SMS initialization error
E2010005	GSM SMS read all unread SMS error
E2010006	GSM SMS read all read SMS error

For other possible general return codes refer General Return Codes section.

2.4 CAN APIs

2.4.1 can_init

can_init	
Description	This API initializes CAN interface.
Syntax	int can_init (const char *name, int bitrate)
Arguments	▪ const char *name – string contains CAN interface name (Ex: can0, can1 or can2) for initializing CAN. ▪ int bitrate – variable which holds the baudrate to be set to the CAN interface.

Return	Refer CAN specific return codes section

Example	char name = “can0”;
	int bitrate = 500000;
	ret = can_init (name, bitrate);

NOTE	▪ To use CANFD(CAN2) follow the below steps ▪ Insert the CANFD module by using below command : “insmod /iwtest/kernel-module/tcan4x5x.ko” ▪ Call can_init () for initialisation.

Below is the software BSP reference for CAN interfaces mapping to the hardware pin-outs.

PIN No	Signal Name (P1)	Description
1	HS_CAN2_H/LS_CAN_H*	Software BSP reference is CAN1
3	FD_CAN_H	Software BSP reference is CAN2
10	HS_CAN2_L/LS_CAN_L*	Software BSP reference is CAN1
12	FD_CAN_L	Software BSP reference is CAN2
13	HS_CAN1_H	Software BSP reference is CAN0
14	HS_CAN1_L	Software BSP reference is CAN0

2.4.2 can_write

can_write	
Description	This API is used to send CAN request with CAN ID with required mode and PID to the can bus.
Syntax	int can_write (char *name, char *data)
Arguments	▪ char *data – data to be written to the can bus. ▪ char *name – can interface name. Eg: “can0”
Return	Refer CAN specific return codes section
Example	char name [] = “can0”; char data [] = “123#AABBCCDD”; ret = can_write (name, data);
NOTE	▪ can_init () should be called before calling can_write () API.

2.4.3 can_read

can_read	
Description	This API is used to read can response messages for the requested CAN PIDs if any request is sent. If no request sent then broadcast messages will be receiving.
Syntax	int can_read (char *name, struct canfd_frame *frame)
Arguments	▪ char *name – can interface name. Eg: “can0” ▪ struct canfd_frame *frame – to store received CAN response data.
Return	Length of CAN data
Example	char name [] = “can0”; struct canfd_frame frame; ret = can_read (name, &frame);
NOTE	▪ can_init () should be called before calling can_read () API.

2.4.4 config_can_wakeup

config_can_wakeup	
Description	This API is used to enable or disable CAN wakeup.
Syntax	int config_can_wakeup (int option)
Arguments	int option – option to enable or disable CAN wakeup 1 – enable 0 – disable
Return	Refer CAN specific return codes section.
Example	int option = 1; ret = config_can_wakeup(option);
NOTE	▪ can_init () should be called before calling this API.

2.4.5 can_deinit

can_deinit	
Description	This API de-initializes the CAN interface.
Syntax	int can_deinit (const char *name)
Arguments	▪ const char *name – string contains CAN interface name (Ex: can0 or can1) for

	deinitializing CAN.
Return	Refer CAN specific return codes section.
Example	char name [] = “can0”; ret = can_deinit(name);
NOTE	▪ can_init () should be called before calling this API.

2.5 CAN specific return codes

Return Codes	Description
90010001	CAN baud rate set error
90010002	CAN initialization error
90010003	CAN interface not found
90010004	CAN Semaphore initialization error
90010005	CAN de-initialization error
90010006	CAN Wakeup enabled error
90010007	CAN0 Wakeup file close error
90010008	CAN1 Wakeup file close error
90010009	CAN read timeout
9001000A	CAN invalid bitrate

For other possible general return codes refer General Return Codes section.

2.6 Ethernet APIs

2.6.1 eth_init

eth_init	
Description	This API initializes Ethernet interface.
Syntax	int eth_init ()
Arguments	None
Return	Refer Ethernet specific return codes section.
Example	ret = eth_init ()
NOTE	-

2.6.2 eth_deinit

eth_deinit	
Description	This API de-initializes Ethernet interface.
Syntax	int eth_deinit ()
Arguments	None
Return	Refer Ethernet specific return codes section.
Example	ret = eth_deinit ()
NOTE	-

2.7 Ethernet specific return codes

Return Codes	Description
60010001	ETH0 MAC Address initialization error
60010002	ETH1 MAC Address initialization error
60010003	Ethernet interface initialization error
60010004	Ethernet interface de-initialization error

For other possible general return codes refer General Return Codes section.

2.8 Accelerometer APIs

2.8.1 acc_init

acc_init	
Description	This API initializes accelerometer.
Syntax	int acc_init ()
Arguments	None
Return	Refer Accelerometer specific return codes section.
Example	ret = acc_init ()
NOTE	-

2.8.2 acc_deinit

acc_deinit	
Description	This API de-initializes accelerometer.
Syntax	int acc_deinit ()
Arguments	None
Return	Refer Accelerometer specific return codes section.
Example	ret = acc_deinit ()
NOTE	-

2.8.3 accelerometer_read

accelerometer_read	
Description	This API reads the accelerometer values from the OBDII device, and returns x-axis, y-axis and z-axis data in the provided structure. The output data will be in unit m/s² (meter per second square). **acc->x** holds x-axis value, **acc->y** holds y-axis value and **acc->z** holds z-axis values.
Syntax	int accelerometer_read (accelerometer_api_priv *acc)
Arguments	▪ accelerometer_api_priv *acc – pointer to structure accelerometer_api_priv
Return	Refer Accelerometer specific return codes section.
Example	accelerometer_api_priv g_adata; ret = accelerometer_read(&g_adata);
NOTE	▪ acc_init () must be called before calling this API.

2.8.4 set_acc_low_pass_filter

set_acc_low_pass_filter	
Description	This API is used to change the low pass filter of accelerometer sensor.
Syntax	int set_acc_low_pass_filter (uint8_t value)
Arguments	▪ uint8_t value – low pass filter value to be written.
Return	Refer Accelerometer specific return codes section.
Example	ret = set_acc_low_pass_filter(0x80);
NOTE	▪ acc_init () must be called before calling this API.

	▪ The value passed to the API should in hex format. ▪ For set_acc_low_pass_filter pass 0x80 as a default value.

2.8.5 set_acc_sampling_frequency

set_acc_sampling_frequency	
Description	This API is used to change the sampling frequency of accelerometer sensor.
Syntax	int set_acc_sampling_frequency (uint8_t value)
Arguments	▪ uint8_t value – sampling frequency value to be written.
Return	Refer Accelerometer specific return codes section.
Example	ret = set_acc_sampling_frequency(0x70);
NOTE	▪ acc_init () must be called before calling this API. ▪ The value passed to the API should in hex format. Refer **Figure 1** for supported frequencies and corresponding values.

CTRL1_XL (10h)

Linear acceleration sensor control register 1 (r/w).

Table 50. CTRL1_XL register

ODR_XL3	ODR_XL2	ODR_XL1	ODR_XL0	FS_XL1	FS_XL0	LPF1_BW_SEL	BW0_XL

Table 51. CTRL1_XL register description

ODR_XL [3:0]	Output data rate and power mode selection. Default value: 0000 (see *Table 52*).
FS_XL [1:0]	Accelerometer full-scale selection. Default value: 00. (00: ±2 *g*; 01: ±16 *g*; 10: ±4 *g*; 11: ±8 *g*)
LPF1_BW_SEL	Accelerometer digital LPF (LPF1) bandwidth selection. For bandwidth selection refer to *CTRL8_XL (17h)*.
BW0_XL	Accelerometer analog chain bandwidth selection (only for accelerometer ODR ≥ 1.67 kHz). (0: BW @ 1.5 kHz; 1: BW @ 400 Hz)

Table 52. Accelerometer ODR register setting

ODR_XL3	ODR_XL2	ODR_XL1	ODR_XL0	ODR selection [Hz] when XL_HM_MODE = 1	ODR selection [Hz] when XL_HM_MODE = 0
0	0	0	0	Power-down	Power-down
1	0	1	1	1.6 Hz (low power only)	12.5 Hz (high performance)
0	0	0	1	12.5 Hz (low power)	12.5 Hz (high performance)
0	0	1	0	26 Hz (low power)	26 Hz (high performance)
0	0	1	1	52 Hz (low power)	52 Hz (high performance)
0	1	0	0	104 Hz (normal mode)	104 Hz (high performance)
0	1	0	1	208 Hz (normal mode)	208 Hz (high performance)
0	1	1	0	416 Hz (high performance)	416 Hz (high performance)
0	1	1	1	833 Hz (high performance)	833 Hz (high performance)
1	0	0	0	1.66 kHz (high performance)	1.66 kHz (high performance)
1	0	0	1	3.33 kHz (high performance)	3.33 kHz (high performance)
1	0	1	0	6.66 kHz (high performance)	6.66 kHz (high performance)
1	1	x	x	Not allowed	Not allowed

Figure 1: Accelerometer ODR setting

- XL_HM_MODE on default is set to 0 that is high performance mode. As per the image we need to change only the CTRL1_XL register based on our requirement.
- LPF1_SW_SEL and BW0_XL is 0 by default.
- Below is the table with the example hex value argument that can be set to the set_acc_sampling_frequency:

Hexa Value	Binary Values							
	ODRXL_3	ODRXL_2	ODRXL_1	ODRXL_0	FSXL_1	FSXL_0	LPF1_BW_SEL	BW0_XL
B0	1	0	1	1	0	0	0	0
70	0	1	1	1	0	0	0	0
78	0	1	1	1	1	0	0	0
7C	0	1	1	1	1	1	0	0
74	0	1	1	1	0	1	0	0

Table 2: Hex values of set_acc_sampling_frequency

2.8.6 set_acc_wakeup_threshold

set_acc_wakeup_threshold	
Description	This API is used to change the wakeup threshold value of accelerometer sensor.
Syntax	int set_acc_wakeup_threshold (uint8_t value)
Arguments	▪ uint8_t value – wake up threshold value to be written.
Return	Refer Accelerometer specific return codes section.
Example	ret = set_acc_wakeup_threshold(0x81);
NOTE	▪ acc_init () must be called before calling this API. ▪ The value passed to the API should in hex format and should be valid.

TAP_THS_6D (59h)

Portrait/landscape position and tap function threshold register (r/w).

Table 185. TAP_THS_6D register

D4D_EN	SIXD_THS 1	SIXD_THS 0	TAP_THS 4	TAP_THS 3	TAP_THS 2	TAP_THS 1	TAP_THS 0

Table 186. TAP_THS_6D register description

D4D_EN	4D orientation detection enable. Z-axis position detection is disabled. Default value: 0 (0: enabled; 1: disabled)
SIXD_THS[1:0]	Threshold for 4D/6D function. Default value: 00 For details, refer to *Table 187*.
TAP_THS[4:0]	Threshold for tap recognition. Default value: 00000 1 LSb corresponds to $FS\_XL/2^5$

Table 187. Threshold for D4D/D6D function

SIXD_THS[1:0]	Threshold value
00	80 degrees
01	70 degrees
10	60 degrees
11	50 degrees

Figure 2: Accelerometer Threshold settings

2.8.7 config_acc_wakeup

config_acc_wakeup	
Description	This API is used to enable or disable accelerometer wakeup.
Syntax	int config_acc_wakeup (int option)
Arguments	▪ int option – option to enable or disable accelerometer wakeup 1 – enable 0 – disable
Return	Refer Accelerometer specific return codes section.
Example	int option = 1; ret = config_acc_wakeup(option);
NOTE	▪ acc_init () must be called before calling this API.

2.9 Accelerometer specific return codes

Return Codes	Description
C0010001	Accelerometer X axis initialization error
C0010002	Accelerometer Y axis initialization error
C0010003	Accelerometer Z axis initialization error
C0010004	Accelerometer buffer initialization error
C0010005	Accelerometer build channel array error
C0010006	Accelerometer scan size error
C0010007	Accelerometer interrupt disable error
C0010008	Accelerometer i2c register configuration error
C0010009	Accelerometer semaphore initialization error

For other possible general return codes refer General Return Codes section.

2.10 Gyroscope APIs

2.10.1 gyro_init

gyro_init	
Description	This function initializes gyroscope.
Syntax	int gyro_init ()
Arguments	None
Return	Refer Gyroscope specific return codes section.
Example	ret = gyro_init ();
NOTE	-

2.10.2 gyro_deinit

gyro_deinit	
Description	This function de-initializes gyroscope.
Syntax	int gyro_deinit ()
Arguments	None
Return	Refer Gyroscope specific return codes section.
Example	ret = gyro_deinit ();

2.10.3 gyroscope_read

gyroscope_read	
Description	This API reads the gyroscope values from the OBDII device, and returns x-axis, y-axis and z-axis data in the provided structure. The output data will be in unit RPS(radian per second). **g_data->x** holds x-axis value, **g_data->y** holds y-axis value and **g_data->z** holds z-axis values.
Syntax	int gyroscope_read (gyroscope_api_priv *g_data)
Arguments	▪ gyroscope_api_priv *g_data – pointer to the structure gyroscope_api_priv
Return	Refer Gyroscope specific return codes section.
Example	gyroscope_api_priv g_data; ret = gyroscope_read(&g_data);
NOTE	▪ gyro_init () must be called before calling this API.

2.10.4 set_gyro_sampling_frequency

set_gyro_sampling_frequency	
Description	This API is used to change the sampling frequency of gyroscope sensor.
Syntax	int set_gyro_sampling_frequency (uint8_t value)
Arguments	▪ uint8_t value – sampling frequency value to be written.
Return	Refer Gyroscope specific return codes section.
Example	ret = set_gyro_sampling_frequency (0x70);
NOTE	▪ gyro_init () must be called before calling this API. ▪ The value passed to the API should in hex format. Refer **Figure 3** for supported frequencies and corresponding values.

9.14 CTRL2_G (11h)

Angular rate sensor control register 2 (r/w).

Table 53. CTRL2_G register

ODR_G3	ODR_G2	ODR_G1	ODR_G0	FS_G1	FS_G0	FS_125	0(1)

1. This bit must be set to '0' for the correct operation of the device.

Table 54. CTRL2_G register description

ODR_G [3:0]	Gyroscope output data rate selection. Default value: 0000 (Refer to *Table 55*)
FS_G [1:0]	Gyroscope full-scale selection. Default value: 00 (00: 250 dps; 01: 500 dps; 10: 1000 dps; 11: 2000 dps)
FS_125	Gyroscope full-scale at 125 dps. Default value: 0 (0: disabled; 1: enabled)

Table 55. Gyroscope ODR configuration setting

ODR_G3	ODR_G2	ODR_G1	ODR_G0	ODR [Hz] when G_HM_MODE = 1	ODR [Hz] when G_HM_MODE = 0
0	0	0	0	Power down	Power down
0	0	0	1	12.5 Hz (low power)	12.5 Hz (high performance)
0	0	1	0	26 Hz (low power)	26 Hz (high performance)
0	0	1	1	52 Hz (low power)	52 Hz (high performance)
0	1	0	0	104 Hz (normal mode)	104 Hz (high performance)
0	1	0	1	208 Hz (normal mode)	208 Hz (high performance)
0	1	1	0	416 Hz (high performance)	416 Hz (high performance)
0	1	1	1	833 Hz (high performance)	833 Hz (high performance)
1	0	0	0	1.66 kHz (high performance)	1.66 kHz (high performance)
1	0	0	1	3.33 kHz (high performance	3.33 kHz (high performance)
1	0	1	0	6.66 kHz (high performance	6.66 kHz (high performance)
1	0	1	1	Not available	Not available

Figure 3: Gyroscope ODR settings

2.10.5 set_gyro_low_pass_filter

set_gyro_low_pass_filter	
Description	This API is used to change the low pass filter of gyroscope sensor.
Syntax	int set_gyro_low_pass_filter (uint8_t value)
Arguments	▪ uint8_t value – low pass filter value to be written.
Return	Refer Gyroscope specific return codes section.
Example	ret = set_gyro_low_pass_filter(0x80);
NOTE	▪ gyro_init () must be called before calling this API. ▪ The value passed to the API should in hex format.

2.11 Gyroscope specific return codes

Return Codes	Description
B0010001	Gyroscope X axis initialization error
B0010002	Gyroscope Y axis initialization error
B0010003	Gyroscope Z axis initialization error
B0010004	Gyroscope buffer initialization error
B0010005	Gyroscope build channel array error
B0010006	Gyroscope scan size error
B0010007	Gyroscope i2c register configuration error
B0010008	Gyroscope semaphore initialization error

For other possible general return codes, refer General Return Codes section.

2.12 Magnetometer APIs

2.12.1 mag_init

mag_init	
Description	This function initializes magnetometer.
Syntax	int mag_init ()
Arguments	None
Return	Refer magnetometer specific return codes section.
Example	ret = mag_init ();
NOTE	-

2.12.2 magnetometer_read

magnetometer_read	
Description	This API reads the magnetometer values from the OBDII device, and returns x-axis, y-axis and z-axis data in the provided structure. **mag->x** holds x-axis value, **mag->y** holds y-axis value and mag->**z** holds z-axis values.
Syntax	int magnetometer_read (magnetometer_api_priv *mag)
Arguments	▪ magnetometer_api_priv *g_data – pointer to the structure magnetometer_api_priv

Return	Refer magnetometer specific return codes section.
Example	magnetometer_api_priv mag; ret = magnetometer_read(&mag);
NOTE	▪ mag_init () must be called before calling this API.

2.12.3 set_mag_sampling_frequency

set_mag_sampling_frequency	
Description	This API is used to change the sampling frequency of magnetometer.
Syntax	int set_mag_sampling_frequency (uint8_t value)
Arguments	▪ uint8_t value – sampling frequency value to be written.
Return	Refer magnetometer specific return codes section.
Example	ret = set_mag_sampling_frequency (0x70);
NOTE	▪ mag_init () must be called before calling this API. ▪ The value passed to the API should in hex format. Refer **Figure 3** for supported frequencies and corresponding values.

2.12.4 mag_deinit

mag_deinit	
Description	This function de-initializes magnetometer.
Syntax	int mag_init ()
Arguments	None
Return	Refer magnetometer specific return codes section.
Example	ret = mag_deinit ();
NOTE	-

2.13 Magnetometer specific return codes

Return Codes	Description
C1000001	Magnetometer build channel array error
C1000002	Magnetometer scan size error

For other possible general return codes, refer General Return Codes section.

2.14 GPS APIs

2.14.1 gps_init

gps_init	
Description	This function initializes GPS.
Syntax	int gps_init ()
Arguments	None
Return	Refer GPS specific return codes section.
Example	ret = gps_init ();
NOTE	▪ gsm_modem_on () has to be called before gps_init () and agps_init () can be called before calling gps_init () for the faster fix.

2.14.2 gps_deinit

gps_deinit	
Description	This function de-initializes GPS.
Syntax	int gps_deinit ()
Arguments	None
Return	Refer GPS specific return codes section.
Example	ret = gps_deinit ();
NOTE	-

2.14.3 agps_init

agps_init	
Description	This function initializes AGPS.
Syntax	int agps_init ()
Arguments	None
Return	Refer GPS specific return codes section.
Example	ret = agps_init ();
NOTE	▪ This API fetches latest extra data from the server. So, this function may take some time.

	▪ Before AGPS init: (Both BG96 and EG95) o In case init (0) is called for device initialization, then establish_connection () API to be called. o In case of init (1), use check_network_connection () API to check PPP link is Up. ▪ **After AGPS init:** o **In case of BG96 Module if init (0) is called establish_connection () API to be called.** o **In case of init (1) Network manager thread will take care of PPP link.** ▪ In case of EG95 series module no need to call establish_connection () API after agps_init ().

2.14.4 get_gps_data

get_gps_data	
Description	This API reads the gps data from the OBDII device and returns the data in the character array recv_data.
Syntax	int get_gps_data (char * nmea, size_t * g_nbytes, char *recv_data, int length)
Arguments	▪ char * nmea – type of nmea message to be read. Eg: “GPRMC”, “GPGSA” etc. ▪ size_t * g_nbytes – pointer to variable which contains number of bytes read ▪ char *recv_data – pointer to character array (128 bytes) to store the received data ▪ int length – Size/Length of the character array to store the received data
Return	Refer GPS specific return codes section.
Example	char recv_data [200] = {0}; Size_t len = 0; ret = get_gps_data (“GPRMC”, &len, recv_data, sizeof(recv_data));
NOTE	▪ NMEA message will be stored to the recv_data buffer. gps_init () must be called before calling this API.

2.15 GPS specific return codes

Return Codes	Description
A0010001	GPS data port initialization error
A0010002	GPS data port de-initialization error
A0010003	GPS data port disconnection error
A0010004	AGPS enable error
A0010005	AGPS set time error
A0010006	AGPS delete existing data fail
A0010007	AGPS HTTP configuration enable fail
A0010008	AGPS QIACT enable error
A0010009	AGPS URL size error
A1010001	AGPS downloaded data invalid
A1010002	AGPS read data invalid
A1010003	AGPS load data invalid
A1010004	AGPS Semaphore initialization error
A1010005	AGPS QICSGP error
A1010006	AGPS listing existing data error
A1010007	AGPS QNTP error
A1010008	GPS port already exist

For other possible general return codes, refer General Return Codes section.

2.16 Battery APIs

2.16.1 i_battery_init

i_battery_init	
Description	This function initializes Battery.
Syntax	int i_battery_init ()
Arguments	None
Return	Refer Battery specific return codes section.
Example	ret = i_battery_init ();

NOTE	▪ This API fetches latest extra data from the server. So, this function may take some time.

2.16.2 i_get_battery_status

i_get_battery_status	
Description	This API is used to check the battery charging status.
Syntax	int i_get_battery_status (int *b_chrg_status)
Arguments	▪ int *b_ chrg_status – Variable to store the battery status
Return	▪ b_chrg _status = 0x1003, if battery not connected ▪ b_chrg _status = 0x1004, if battery fully charged ▪ b_chrg _status = 0x1005, if battery is charging ▪ b_chrg _status = 0x1006, if battery is not charging ▪ b_chrg _status = 0x1007, if battery is unstable ▪ Refer Battery specific return codes section.
Example	int batter_chrg_status; ret = i_get_battery_status (& batter_chrg_status);
NOTE	▪ If the battery charge status is being shown as UNSTABLE_STATE(0x1007) but the proper voltage is displayed and battery heath is good, it implies the battery is in Shipment mode. ▪ Battery charge status will be in UNSTABLE STATE (0x1007) if battery charging is disabled.

2.16.3 i_battery_get_voltage

i_battery_get_voltage	
Description	This API is used to check the battery voltage.
Syntax	int i_battery_get_voltage (double *i_bat_volt)
Arguments	▪ double *i_bat_volt – address of variable to store voltage value.
Return	Refer Battery specific return codes section.
Example	double i_bat_volt; ret = i_battery_get_voltage(&i_bat_volt);

NOTE	▪ i_bat_volt contains the battery voltage value.

2.16.4 i_battery_get_health

i_battery_get_health	
Description	This API is used to check the battery health status.
Syntax	int i_battery_get_health ()
Arguments	None
Return	▪ 0x1001, if battery is in good health. ▪ 0x1002, if battery is not in good health. ▪ Refer Battery specific return codes section.
Example	ret = i_battery_get_health ();
NOTE	▪ i_bat_volt contains the battery voltage value.

2.16.5 battery_connect_config

battery_connect_config	
Description	This API is used to either disable/enable shipment mode. (Shipment mode will disable the power flow from the battery (similar to disconnecting the battery from the device even though it is connected to the device)).
Syntax	int battery_connect_config (int con_status)
Arguments	▪ int con_status: con_status either low/high
Return	Refer Battery specific return codes section.
Example	int con_status = 0; ret = battery_connect_config(con_status);
NOTE	▪ If low (0) is passed, shipment mode will be enabled (i.e., disabling power flow from the battery (similar to disconnecting the battery from the device even though it is connected to the device)). ▪ If high (1) is passed, shipment mode will be disabled (i.e., enabling all the functionalities related to battery (charging, discharging and other functionalities)).

2.16.6 battery_charge_state_config

battery_charge_state_config	
Description	This API is used to either disable/enable battery charging.
Syntax	int battery_charge_state_config (int state)
Arguments	▪ int state: state has to be either low/high
Return	Refer Battery specific return codes section.
Example	int state=1; ret = battery_charge_state_config (state);
NOTE	▪ If low (0) is passed, battery charging will be enabled. ▪ If high (1) is passed, battery charging will be disabled.

2.16.7 get_power_source

get_power_source	
Description	This API is used to check the power source status.
Syntax	int get_power_source ()
Arguments	None
Return	▪ 0x1008, if connected to external power ▪ 0x1009, if connected to internal power ▪ Refer Battery specific return codes section.
Example	ret = get_power_source ();
NOTE	**Sequence to be followed for battery** ▪ init (). ▪ battery_init (). ▪ Any API related to battery.

2.17 Battery specific return codes

Return Codes	Description
D0010001	Battery voltage node open error
D0010002	Battery voltage node read error
D0010003	Battery health node open error

D0010004	Battery health node read error
D0010005	Battery not connected
D0010006	Battery is fully connected
D0010007	Battery is charging
D0010008	Battery is not charging
D0010009	Battery unstable state

For other possible general return codes, refer General Return Codes section.

2.18 WiFi APIs

2.18.1 wifi_init

wifi_init	
Description	This API is used to initialize WiFi interface with hostapd or station mode.
Syntax	int wifi_init (int mode)
Arguments	int mode – enable hostapd or station mode. 1 – hostapd mode 0 – station mode
Return	Refer WiFi specific return code section.
Example	int mode = 0; ret = wifi_init(mode);
NOTE	▪ The WiFi on the OBDII device is configured to be enabled automatically (using wifi_enable.service) on device boot up to enable easy file transfer. Trying to initialize it again will lead to error. (0x81010003-Wi-Fi has already been initialised and is active) ▪ In-order to avoid getting the above error code and to avoid enabling Wi-Fi on boot up, service needs to be disabled or modified. ▪ In-order to enable routing of the 4G internet connection through the Wi-Fi interface, please ensure that 4G is initialized with an internet connection before initializing the Wi-Fi interface. ▪ Routing 4G internet connection through the Wi-Fi interface can be availed only

	when the Wi-Fi is initialized in Access Point Mode.

2.18.2 wifi_deinit

wifi_deinit	
Description	This API is used to deinitialize WiFi interface with hostapd or station mode.
Syntax	int wifi_deinit (int mode)
Arguments	int mode – disable hostapd or station mode. 1 – hostapd mode 0 – station mode
Return	Refer WiFi specific return code section.
Example	int mode = 0; ret = wifi_deinit(mode);
NOTE	-

2.19 WiFi specific return codes

Return Codes	Description
80010001	WiFi interface initialization error
80010002	WiFi interface deinitialization error
80010003	WiFi interface Up error
80010004	WiFi hostapd mode initialization error
80010005	WiFi station mode initialization error
80010006	WiFi hostapd mode deinitialization error
80010007	WiFi station mode deinitialization error
80010008	WiFi IP set error
80010009	WiFi hostapd mode udhcpd configuration error

For other possible general return codes, refer General Return Codes section.

2.20 Bluetooth APIs

2.20.1 ble_init

ble_init	
Description	This API is used to initialize Bluetooth Interface.
Syntax	int ble_init ()
Arguments	None
Return	Refer Bluetooth specific return codes section.
Example	ret = ble_init ();
NOTE	-

2.20.2 ble_deinit

ble_deinit	
Description	This API is used to de-initialize Bluetooth Interface.
Syntax	int ble_deinit ()
Arguments	None
Return	Refer Bluetooth specific return codes section.
Example	ret = ble_deinit ();
NOTE	-

2.21 Bluetooth specific return codes

Return Codes	Description
70010001	Bluetooth interface initialization error
70010002	Bluetooth interface deinitialization error

For other possible general return codes, refer General Return Codes section.

3 General Return Codes

Error Number	Description
0	Success
-1	Failure
-2	Invalid argument passed
100A	Invalid baudrate passed
F0010001	Invalid GPIO number
F0010002	GPIO export file open error
F0010003	GPIO export file write error
F0010004	GPIO direction file open error
F0010005	GPIO direction file write error
F0010006	No access to GPIO direction file
F0010007	GPIO value file open error – to read value
F0010008	GPIO value file read error
F0010009	No access to GPIO value file – to read value
F0010010	GPIO value file open error – to write value
F1010001	GPIO value file write error
F1010002	No access to GPIO value file – to write value
F1010003	GPIO read file open error
F1010004	Invalid GPIO read
F1010005	No access to GPIO file
F1010006	GPIO event number read error
F2010001	Invalid serial port
F2010002	Serial port initialization error
F2010003	Serial port write error
F2010004	Serial port read error
F2010005	Serial port baud rate set error
F2010006	Serial port deinitialization error
F3010001	Invalid i2c bus

F3010002	I2c file open error
F4010001	Error in reading MAC address
F4010002	Error in accessing interface
F4010003	Error reading the file

4 I2C Bus Number for Devices

Device Name	Bus Number
Accelerometer	1
Gyroscope	1
Battery	1
Magnetometer	0

Table 2: I2C Bus Number for Devices

5 Structures

This chapter describes structures in the OBDII library.

5.1 _accelerometer_api_priv

```
typedef struct _accelerometer_api_priv {
   int fd;
   double x, y, z, acc;
} accelerometer_api_priv;
```

This structure holds the accelerometer data,

fd = accelerometer device node

x = x-axis value

y = y-axis value

z = z - axis value

5.2 _gyroscope_api_priv

```
typedef struct _gyroscope_api_priv {
  double x, y, z;
} gyroscope_api_priv;
```

This structure holds the gyroscope data,

x = x axis value

y = y axis value

z = z axis value

5.3 _magnetometer_api_priv

```
typedef struct _magnetometer_api_priv {
  double x, y, z;
} magnetometer_api_priv;
```

This structure holds the magnetometer data,

x = x axis value

y = y axis value

z = z axis value

6 API Sequence

Please follow the below sequence in application, while calling the OBDII library API's.

1. init ()
2. i_battery_init
3. gsm_modem_on
4. agps_init
5. gps_init
6. acc_init
7. gyro_init
8. mag_init
9. can_init

Sequence to be followed if 0 is passed to init ()

- i_battery_init ()
- gsm_modem_on ()
- check_gsm_modem_status ()
- establish_connection ()
- check_network_connection ()
- check_gsm_nw_connection ()
- get_gsm_signal_strength ()
- get_gsm_nw_reg ()

SAMPLE API Source

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "common.h"
#include "gps.h"
#include <fcntl.h>
#include <termios.h>
```

```
#include <signal.h>
#include <stdint.h>
#include <linux/netlink.h>
#include <linux/if_link.h>
#include <linux/input.h>
#include <linux/rtnetlink.h>
#include <netdb.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <net/if.h>
#include <unistd.h>
#include <errno.h>
#include "gsm.h"
#include "gyroscope.h"
#include "accelerometer.h"
#include "error_nos.h"
#include "magnetometer.h"
#include "battery.h"

#define ENABLE 1
#define DISABLE 0
#define WIFI_HOSTAPD_MODE 1
#define WIFI_STATION_MODE 2
int sleep_counter = 0;

/ Printing GPS read data /
int gps_data_ap(char buf[200], size_t nbytes)
{
        int j, k, l;
        char field[15][100] = {0};
```

```
        printf("\n GPS-DATA[");
        for(j=0; j<nbytes; j++)
        {
                printf("%c", buf[j]);
        }
        printf("]\n");
        return 0;
}
int main()
{
        int rc = 0;
        int sim_reg_count = 0;
        char arr_sig_strngth[100] = {0};
        char cell_id[50] = {0},lac[30] = {0};
        int sig_strength = 0;
        int arr_sig_strength = 0;
        int network_link_count = 0;
        int network_connection_count = 0;
        int ntp_server_update_count = 0;
        int link_status = 0;
        char iccid[ 64 ] = " ";
        char buf[ 100 ] = " ";
        char resp_buffer[ 1024 ];
        char imei[100] = {0};
        char recv_data[200] = {0};
        int gps_fd;
        size_t len = 0;
        int n = 0;
        int count = 0;
        int battery_chrg_status = 0;
        double battery_voltage = 0;
```

```
accelerometer_api_priv g_adata;
gyroscope_api_priv g_data;
magnetometer_api_priv mag;
int cpuid_length = 40;
char cpu_id[50] = {0};
char wifi_addr[30] = {0},eth_addr[30] = {0}, ble_addr[30] = {0};

init:
        count = 0;
        rc = init(0);
        if(rc == -1)
                printf("init() failed\n");
        else
                printf("**********************init() done\n");
        rc = i_battery_init( );
        printf("i_battery_init rc value %x\n", rc );
        memset( cpu_id, 0, sizeof( cpu_id ) );
        rc = get_cpu_id(cpu_id, cpuid_length);
        printf("get_cpu_id rc value %x\n", rc );
        printf("get_cpu_id buffer %s\n", cpu_id );

        /*Initializing ethernet interface */
        rc = eth_init( );
        printf("eth_init rc value %x\n", rc );

        / Getting Bluetooth MAC address /
        get_mac_address("eth0", eth_addr);
        printf("ETHERNET MAC ADDRESS %s\n", eth_addr);

        /*Deinitializing ethernet interface */
        rc = eth_deinit( );
```

```
printf("eth_deinit rc value %x\n", rc );

/*Initializing Bluetooth interface */
rc = ble_init( );
printf("ble_init rc value %x\n", rc );

/* Getting Bluetooth MAC address */
get_mac_address("hci0",ble_addr);

/*Initializing WiFi interface */
rc = wifi_init( WIFI_HOSTAPD_MODE );
printf("wifi_init rc value %x\n", rc );

/* Getting MAC address of WiFi interface */
get_mac_address("wlan0",wifi_addr);

printf("\n#############[GSM Modem]###############\n");
/ checking whether modem is on or off. /
rc = check_gsm_modem_status();
if(rc != 0)
{
/ Turning on the GSM modem /
        rc = gsm_modem_on( "0000", 4 );
        printf("gsm_modem_on() Return value = %x\n",rc);
}
printf("\n LINE %d\n", __LINE__);
/* Getting the IMEI number of the GSM module. */
memset( imei, 0, sizeof( imei ) );
rc = get_gsm_imei(imei, sizeof( imei ) );
printf("imei[%s] rc = %x\n", imei, rc);
```

```
/* Getting the ICCID of the GSM module */
memset( iccid, 0, sizeof( iccid ) );
rc = get_gsm_sim_iccid(iccid, sizeof(iccid));
printf("iccid[%s] = %x\n", iccid, rc);

/* Getting the signal range details of the network */
memset( arr_sig_strngth, 0, sizeof( arr_sig_strngth ) );
rc = get_gsm_signal_strength(arr_sig_strngth, sizeof( arr_sig_strngth ));
printf("gsm signal[%s] = %x\n", arr_sig_strngth, rc);

/* Getting the SIM registration details */
memset( cell_id, 0, sizeof( cell_id ) );
memset( lac, 0, sizeof( lac ) );
rc = get_gsm_nw_reg(cell_id, sizeof( cell_id ), lac, sizeof( lac ) );
printf("CELL_ID %s LAC %s\n", cell_id, lac);

/* Changing the network mode */
rc = set_gsm_network_mode(4);
printf("rc set_gsm_network_mode() %x \n", rc);

/* Sending AT commands and receive the response */
gsm_apn_configuration("airtelgprs.com","ATDT*99***1#","abcd","1234");
while (1) {
/*Establishing ppp network connection */
        rc = establish_connection();
        if(rc == 0) {
        /* Checking whether the internet connectivity is active or not */
                rc = check_network_connection ();
                if (rc == 0) {
                        rc = check_gsm_nw_connection ();
                        break;
```

```
                }
                else{
                        printf("Trying to Connect!!!!\n");
                        sleep(1);
                        if (count > 2){
                        printf("Network Connection Not Established\n");
                        break;
                        }
                }
        }
        else
        {
                sleep(1);
                if (count >= 2 ){
                        printf("Network Connection Not Established\n");
                        break;
                }
        }
        count++;
}
/* Updating device time to standard UTC time using ntp time server */
rc = ntp_server_update();
if (rc == -1){
        printf("NTP FAILED rc %x\n", rc);
}
else
        printf("NTP DONE %x\n", rc);
n = 0;
printf("\n#############[LED]##############\n");
/ Turning on the green LED /
rc = led_enable();
```

```
printf("led_enable() Return value = %x\n", rc);
printf("\n#############[GPS]###############\n");
/ Fetching latest xtradata from the server /
rc = agps_init();
printf("quectel_agps_init() Return value = %x\n", rc);

/* Initializing gps */
rc = gps_init();
printf("gps_init() Return value = %x\n", rc );

/* Reading the gps data from the OBDII device */
memset(recv_data, 0, sizeof( recv_data ) );
get_gps_data("GPRMC", &len, recv_data, sizeof( recv_data));
printf("get_gps_data len value %d\n", len);
printf("get_gps_data recv_data value %s\n", recv_data);
gps_data_ap(recv_data, len);
rc = GSM_set_to_message_init ( );
printf("GSM_set_to_message_init rc value is %x\n", rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
rc = unread_message( resp_buffer, sizeof( resp_buffer ), 800000 );
printf("unread_message resp_buffer %s and rc value %x\n", resp_buffer, rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
rc = read_message( resp_buffer, sizeof( resp_buffer ), 800000 );
printf("read_message resp_buffer %s and rc value %x\n", resp_buffer, rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
rc = delete_message( 1, 800000 );
printf("delete_message rc value %x\n", rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
rc = unread_message( resp_buffer, sizeof( resp_buffer ), 800000 );
printf("unread_message resp_buffer %s and rc value %x\n", resp_buffer, rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
```

```
rc = read_message( resp_buffer, sizeof( resp_buffer ), 800000 );
printf("read_message resp_buffer %s and rc value %x\n", resp_buffer, rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
rc = delete_all_messages( 800000 );
printf("delete_all_message rc value %x\n", rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
rc = unread_message( resp_buffer, sizeof( resp_buffer ), 800000 );
printf("unread_message resp_buffer %s and rc value %x\n", resp_buffer, rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
rc = read_message( resp_buffer, sizeof( resp_buffer ), 800000 );
printf("read_message resp_buffer %s and rc value %x\n", resp_buffer, rc );
memset( resp_buffer, 0, sizeof( resp_buffer ) );
rc = send_sms("Hii, I have an user application", "7907803205", 800000 );
printf("send_sms resp_buffer %s and rc value %x\n", resp_buffer, rc );
rc = network_monitor_disable( );
printf("network_monitor_disable rc value is %x\n", rc );
rc = set_gsm_flight_mode_on();
printf("set_gsm_flight_mode_on() return value %x\n", rc);
sleep(2);
rc = set_gsm_flight_mode_off();
printf("set_gsm_flight_mode_off() return value %x\n", rc);

/ De-initializing gps /
rc = gps_deinit( );
printf("gps_deinit rc value 0x%x\n", rc );
rc = gsm_modem_off();
printf("gsm_modem_off rc value 0x%x\n", rc );
while(1)
{
        n = n+1;
        / Checking the battery charging status /
```

```
                rc = i_get_battery_status( &battery_chrg_status );
                printf("***************battery Status ************** = %x\n", rc);
                sleep(1);
                if(n>=10)
                        break;
        }
        rc = i_battery_get_voltage( &battery_voltage );
        printf("i_battery_get_voltage rc value 0x%x and battery_voltage %f\n", rc,
        battery_voltage );
        rc = i_battery_get_health( );
        printf("i_battery_get_health rc value 0x%x\n", rc );
        rc = i_get_battery_status( &battery_chrg_status );
        printf("i_get_battery_status rc value %x and battery_chrg_status %d\n", rc,
        battery_chrg_status );

        printf("\n#############[Accelerometer]##############\n");
        / Initializing accelerometer /
        rc = acc_init();
        printf("\n Accelerometer Initialisation return value = 0x%x\n", rc);
        n =0;
        while(1)
        {
                n = n+1;
                / Reading the accelerometer values from the OBDII device /
                memset( &g_adata, 0, sizeof( g_adata ) );
                accelerometer_read(&g_adata);
                printf ("Acc x-axis: %f\ty-axis: %f\tz-axis: %f\n", g_adata.x,g_adata.y,
g_adata.z);
                if(n >= 10)
                        break;
        }
```

```
printf("\n#############[Gyroscope]###############\n");
/ Initializing gyroscope /
rc = gyro_init();
printf("\n Gyroscope Initialisation return value = 0x%x\n", rc);
n =0;
while(1)
{
        n = n+1;
        memset( &g_data, 0, sizeof( g_data ) );
        / Reading the gyroscope values from the OBDII device /
        gyroscope_read(&g_data);
        printf ("Gyro x-axis: %f\ty-axis: %f\tz-axis: %f\n",g_data.x, g_data.y,
g_data.z);
        if(n >= 10)
                break;
}
rc = set_acc_wakeup_threshold( 0x89 );
printf("set_acc_wakeup_threshold rc value 0x%x\n", rc );
set_acc_sampling_frequency(0x70);
set_acc_low_pass_filter(0x80);
set_acc_wakeup_threshold(0x81);
set_gyro_sampling_frequency(0x70);
rc = config_timer_wakeup( ENABLE, 60 );
printf("config_timer_wakeup rc value %x\n", rc );
rc = config_acc_wakeup( ENABLE );
printf("config_acc_wakeup rc value %x\n", rc );
rc = config_ignition_wakeup( ENABLE );
printf("config_ignition_wakeup rc value %x\n", rc );
rc = config_can_wakeup( ENABLE );
printf("config_can_wakeup rc value %x\n", rc );
```

```
while( 1 )
{
        rc = network_monitor_disable( );
        printf("network_monitor_disable rc value is %x\n", rc );
        if( rc == 0 )
        {
                break;
        }
        else
        {
                sleep( 1 );
        }
}
/* De-initializing gyroscope */
rc = gyro_deinit ();
printf("gyro_deinit rc value %x\n", rc );
/* De-initializing accelerometer */
rc = acc_deinit ();
printf("acc_deinit rc value %x\n", rc );
/* De-initializing bluetooth */
rc = ble_deinit( );
printf("ble_deinit rc value %x\n", rc );
/* De-initializing wifi */
rc = wifi_deinit( WIFI_HOSTAPD_MODE );
printf("wifi_deinit rc value %x\n", rc );
printf("\n#############[LED]###############\n");
/* Turning off the green LED */
rc = led_disable();
printf("led_disable() Return value = %x\n", rc);
printf("###################DE-INIT#######################\n");
rc = deinit( );
```

```
            if(rc == -1)
                printf("deinit() failed\n");
            else
                printf("**********deinit() done\n");
                rc = push_device_to_sleep( );
                printf("push_device_to_sleep rc value %x\n", rc );
                sleep_counter++;
                printf("Sleep counter value %d\n", sleep_counter );
                if( sleep_counter > 5 )
                {
                    printf("Sleep counter reached more than 5 ############\n" );
                    rc = restart_device();
                    printf("restart_device rc value %x\n", rc );
                }
            goto init;
}
```

7 Application development

To develop an application for OBDII device please follow the below steps,

Headers

Include the **accelerometer.h**, **battery.h, common.h, gyroscope.h, magnetometer.h, gps.h, can.h, error_nos.h, gsm.h, and debug.h** header files to the application.

Library

Link the **libOBD2.so** library to the application using **"-lOBD2**" option during compilation

Cross_Compiler

Follow the below steps to compile the Application for OBD II Device

Step1: Copy the .tar.bz2 (cross-compiler file) file provided to an any directory of choice

Step2: Untar the cross-compiler file using the below command

 # sudo tar -xvjf <cross-compiler-file-name>

Step3: A script file (.sh) file (fsl-imx-x11-glibc-x86_64-imx-image-core-cortexa7t2hf-neon-imx6ull-iwg18m-sm-toolchain-5.4-zeus.sh) will now be obtained in the same directory. Run the below command to install the cross-compiler in the default location (/opt/fsl-imx-x11) in the host PC

 # ./fsl-imx-x11-glibc-x86_64-imx-image-core-cortexa7t2hf-neon-imx6ull-iwg18m-sm-toolchain-5.4-zeus.sh

Step3: Include Cross-Compiler headers to your application by adding below options in your Makefile.

-I /opt/fsl-imx-x11/5.4-zeus/sysroots/x86_64-pokysdk-linux/usr/include/

Step4: Copy the released libOBD2.so library file to Application directory

 # cp libOBD2.so <path to Application folder>

Step5: Link libOBD2.so library to your application by adding below option in your Makefile.

-L <Path To libOBD2.so> -lOBD2

Step6: Copy the libOBD2.so library to the below path in the root file-system in the OBD-II device.

Path : /usr/lib/

Step7: Setup the environment in terminal /shell window before compiling the application using the below command:

 # source /opt/fsl-imx-x11/5.4-zeus/environment-setup-cortexa7t2hf-neon-poky-linux-gnueabi

Technical Support

***i*Wave Systems Technologies Pvt. Ltd.**

7/B, 29th Main,

BTM Layout 2nd Stage,

Bangalore – 560 076

Phone: +91-80-26683700, 26786245

Fax : +91-80-26685200

Email : support.obd@iwavesystems.com

Web : www.iwavesystems.com