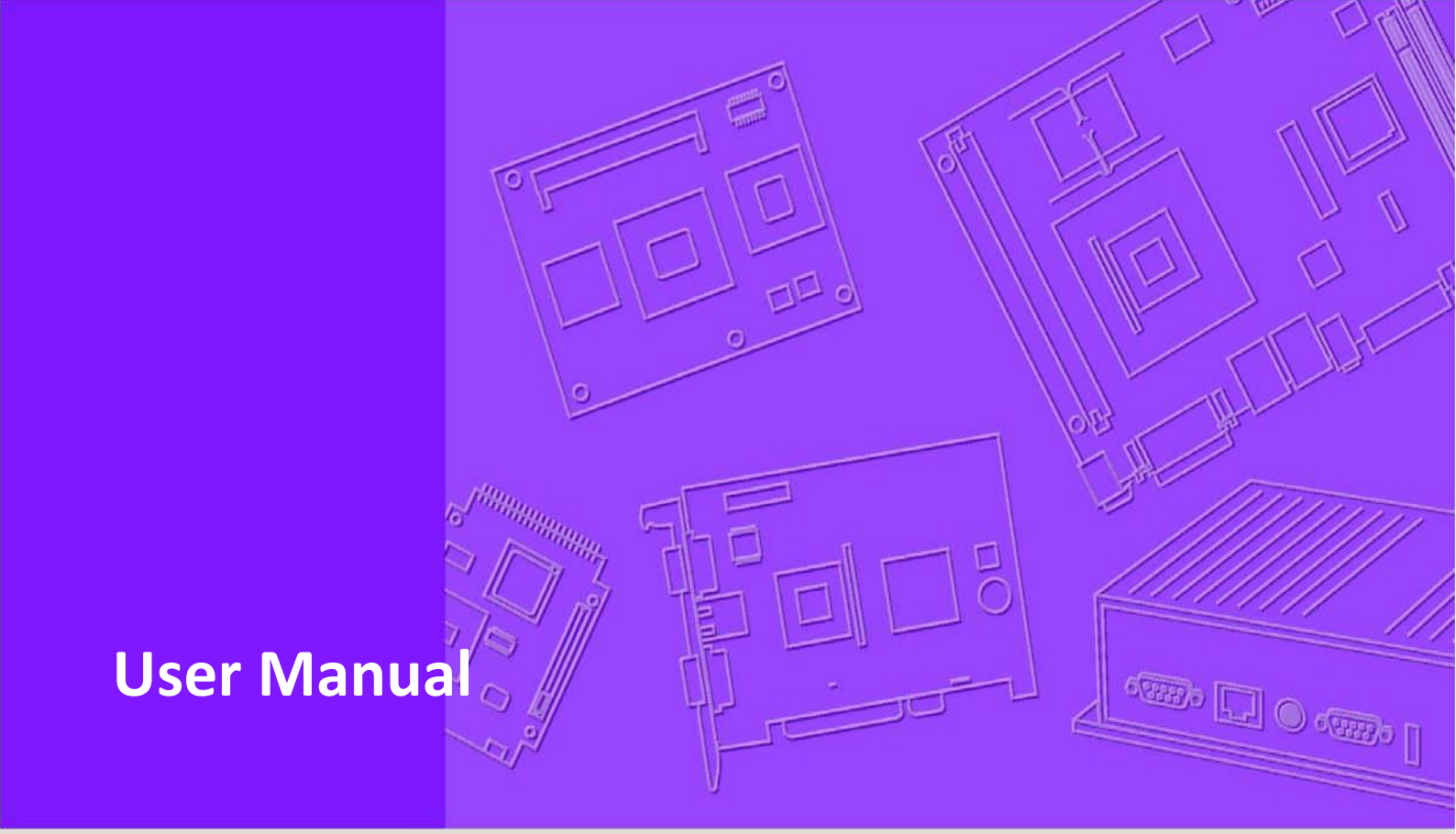

User Manual

SUSI CAN Bus

Secured & Unified Smart Interface Software for CAN Bus APIs

Date: 2019/2/18

Contents

1. Introduction

The SUSI CAN bus demo program demonstrates how to incorporate SUSI CAN bus library into user's own applications. The program is written in C# programming language and based upon .NET Compact Framework 3.5, Visual Studio 2015. If you plan to write your own application you can refer to the source code of the Demo program which contains all functions provided by Advantech SUSI. Contact with your local FAE if you have any question about this.

2. Definition

2.1 SUSI CAN bus ID

- RDC CAN bus hardware

#define SUSI_ID_RDC_CAN0 ..... (0)
#define SUSI_ID_RDC_CAN1 ..... (1)

- APACER CAN bus hardware

#define SUSI_ID_APACER_CAN0 ..... (2)
#define SUSI_ID_APACER_CAN1 ..... (3)

- VCIL CAN bus hardware

#define SUSI_ID_VCIL_CAN0 ..... (4)
#define SUSI_ID_VCIL_CAN1 ..... (5)
#define SUSI_ID_VCIL_CAN2 ..... (6)

2.2 Status/Error code

These error codes are the same as SUSI4.1 definition.
#define SUSI_STATUS_NOT_INITIALIZED .....(0xFFFFFFFF)
#define SUSI_STATUS_INVALID_PARAMETER..... (0xFFFFFEFF)
#define SUSI_STATUS_UNSUPPORTED ..... (0xFFFFFCFF)
#define SUSI_STATUS_READ_ERROR..... (0xFFFFFAFF)
#define SUSI_STATUS_WRITE_ERROR ..... (0xFFFFFAFE)
#define SUSI_STATUS_ERROR.....(0xFFFFF0FF)
#define SUSI_STATUS_SUCCESS ..... (0)

2.3 com_setting_t

- **Syntax:**

```
typedef struct
{
    int IsEventMode;
    int com1_num;
    int com2_num;
} com_setting_t;
```

- Description:

  Set the COM port and read CAN bus data method for event mode or polling mode.

IsEventMode

Only Apacer and vcil support event mode. When Apacer/Vcil CAN bus initialize, set 1 to enable event mode and set 0 to enable polling mode.

Polling mode:

SUSI CAN bus demo application will create a thread to polling the CAN bus raw data.

Event mode:

SUSI CAN bus demo application will create a thread and open event to wait the CAN bus raw data. If application got the event, then application will call SusiCanbusRead() API to read raw data.

com1_num

Apacer and vcil need set COM number to initialize. Apacer can auto get COM port. Vcil need to set COM number. For example COM7 that does enter '7' integer. RDC will not use this parameter.

com2_num

Only Vcil CAN bus need set COM2 number. For example COM8 that does enter '8' integer. RDC and Apacer will not use this parameter.

2.4 vcil_can_message_t

- **Syntax:**

```
typedef struct
{
    unsigned char port;
    char length;
    bool remote_request;
    bool extended_frame;
    unsigned int id;
    unsigned char data[8];
} vcil_can_message_t;
```

- Description:

This data structure defines the CAN message.

- **Parameters**

port

This message come from/send to which port.

length

The standard CAN message data length. This data length should not over 8.

remote request

This field is used to indicate whether this CAN message is a remote transmit request(RTR) frame. The value is 1 if the message is a RTR frame(the RTR field of the CAN message identifier is 1).

The value is 0 if the message is not a RTR frame.

extended frame

This field is used to indicate that the message is a standard format(CAN2.0A) or a extended format(CAN2.0B) message.

The value is 1 if the message is a CAN2.0B message (with 29-bits identifier).

The value is 0 if the message is a CAN2.0A message (with 11-bits identifier).

id

The Identifier of CAN.

data

The data array of CAN message

2.5 vcil_can_speed

- **Syntax**

```
typedef enum vcil_can_speed
{
    VCIL_CAN_SPEED_125K = 0,
    VCIL_CAN_SPEED_250K = 1,
    VCIL_CAN_SPEED_500K = 2,
    VCIL_CAN_SPEED_1M   = 3,
    VCIL_CAN_SPEED_200K = 4,
    VCIL_CAN_SPEED_100K = 5,
    VCIL_CAN_SPEED_800K = 6,
    VCIL_CAN_SPEED_83K = 7,
    VCIL_CAN_SPEED_50K = 8,
    VCIL_CAN_SPEED_20K = 9,
    VCIL_CAN_SPEED_10K = 10,
    VCIL_CAN_SPEED_5K = 11,
} vcil_can_speed;
```

- **Support baud rate list**

Ⓥ: Support

ID	Baud rate	RDC	Apacer	Vcil
0	125K	Ⓥ	Ⓥ	Ⓥ
1	250K	Ⓥ	Ⓥ	Ⓥ
2	500K	Ⓥ	Ⓥ	Ⓥ
3	1M	Ⓥ	Ⓥ	Ⓥ
4	200K		Ⓥ	Ⓥ
5	100K	Ⓥ		Ⓥ
6	800K		Ⓥ	Ⓥ
7	83K			Ⓥ
8	50K	Ⓥ		Ⓥ
9	20K			Ⓥ
10	10K			Ⓥ
11	5K			Ⓥ

2.6 vcil_can_mask_t

- **Syntax:**

```
typedef struct
{
    unsigned char type;
    unsigned char bank;
    bool remote_request;
    bool extended_frame;
    unsigned int id1;
    unsigned int mask1;
    unsigned int id2;
    unsigned int mask2;
} vcil_can_mask_t;
```

- **Description:**

This data structure defines the hardware CAN mask.

- **Parameters**

type

This mask type. This field reserved for future configuration currently ignore.

bank

The mask bank, the VCIL supported maximum 13 bank which 1~13. This bank should not over 13. You can think of the bank is a rule of CAN message hardware filter.

remote request

This field is used to indicate whether this CAN message is a remote transmit request(RTR) frame. The value is 1 if the message is a RTR frame(the RTR field of the CAN message identifier is 1).

The value is 0 if the message is not a RTR frame.

extended frame

This field is used to indicate that the message is a standard format(CAN2.0A) or a extended format(CAN2.0B) message.

The value is 1 if the message is a CAN2.0B message (with 29-bits identifier).

The value is 0 if the message is a CAN2.0A message (with 11-bits identifier).

id1

The Identifier 1 of bank.

mask1

The mask 1 of the bank.

id2

The Identifier 2 of bank.

This field will be ignored if extended frame is set to 1.

mask2

The mask 2 of the bank.

This field will be ignored if extended frame is set to 1.

2.7 vcil_can_error_status

- **Syntax:**

```
typedef struct
{
    unsigned int rec;                // receive error counter
    unsigned int tec;                // transmit error counter
    unsigned int last_error_code; // last error code
    unsigned int error_flag;
} vcil_can_error_status;
```

- **Description:**

This CAN controller error status.

- **Parameters**

rec

Receive error counter

tec

Transmit error counter

last_error_code

Last error code.

error_flag

Only VCIL CAN bus support this parameter.

CAN Bus error flag

2.8 rdc_configuration_t

- **Syntax:**

```
typedef struct
{
    unsigned long BaudRate;
    //Sampling Rule
    unsigned long PropSEG;          // 1~8
    unsigned long PSEG1;            // 1~8
    unsigned long PSEG2;            // 1~8
    unsigned long SJW;              // 1~4
    unsigned long Sampling;         // 1: enable sampling 3 times
    //Initial Clock
    unsigned long ClockHz;
    //Global Control
    unsigned long   bArbitrationLostRetry;  //1
    unsigned long   bBusErrorRetry;         //1
    unsigned long   bPowerSaving;           //0
} rdc_configuration_t;
```

- **Description: (only RDC CAN bus available)**
  In CAN Specification, ArbitrationLostRetry & BusErrorRetry is enabled. If ArbitrationLostRetry is disabled and there is an ArbitrationLost, the hardware will not retry the transmit buffer and report transmit status with ArbitrationLost. The behavior of BusErrorRetry is the same with ArbitrationLostRetry.

- **Refer to CAN Specification for Bus timing**
  (Sampling Rule only RDC CAN bus available) In General, in order to communicating with different vendor's CAN device, App should setup the Bus timing and let the location of sample point of CAN devices on the same bus be close. The location of sample point is located between PhaseSEG1 and PhaseSEG2. The possible combination of PropSEG, PSEG1, PSEG2 is the following table:

- **Parameters**

Parameter	Value	Description
BaudRate	1000000	Default 1M bit/s
PropSEG	Valid value is 1 to 8.	Propagation Segment.
PSEG1	Valid value is 1 to 8.	Phase Segment1.
PSEG2	Valid value is 1 to 8.	Phase Segment2.
SJW	Valid value is 1 to 4. Default:1	Synchronization jump width.
Sampling	1: enable, 0: disable. Default:0	Three time sampling Enable.
ClockHz	SUSI CAN bus set 20MHz as default	Source Clock
bArbitrationLostRetry	1: enable, 0: disable. Default:1	Lost retry
bBusErrorRetry	1: enable, 0: disable. Default:1	Error retry
bPowerSaving	1: enable, 0: disable. Default:0	Power setting

- **Suggestion sampling rule**

Ⓥ: Default setting rule

	Baud Rate (bps)	SJW	PropSEG	PSEG1	PSEG2
Ⓥ	1M	1	1	4	4
Ⓥ	500K	1	3	8	8
	500K	1	1	4	4
Ⓥ	250K	1	3	8	8
	250K	1	1	4	8
	250K	1	1	3	3
Ⓥ	125K	1	3	8	8
	125K	1	1	4	4
	125K	1	3	6	6
	125K	1	1	3	3
Ⓥ	100k	1	3	8	8

	100k	1	1	4	4
	100k	1	8	8	8
Ⓥ	50k	1	3	8	8
	50k	1	1	4	4
	50k	1	1	3	3

2.9 vcil_j1939_message_t

- **Syntax:**

```
typedef struct
{
    unsigned char port;
    unsigned int pgn;
    unsigned char destination;
    unsigned char source;
    unsigned char priority;
    int length;
    unsigned char data[8];
} vcil_j1939_message_t;
```

- **Description:**

This data structure defines the J1939 message.

- **Parameter**

port

This message come from/send to which port.

pgn

The parameter group number of this message. currently only support 0 to 0x1FFFF

destination

Ignore

source

The destination address of this message. 0x00 to 0xFF

priority

The priority of this message. The priority of range is 0 to 7. Normally set to 6.

length

The standard J1939 message data length.

data

The data array of J1939 message.

2.10 vcil_j1939_receive_message_t

- **Syntax:**

```
typedef struct
{
        unsigned int dwID;
        unsigned char bPriority;      // Priority
        unsigned char bDataPage;    // Data Page
        unsigned char bPDUFormat;// PDU Format
        unsigned char bPDUSpecific;// PDU Specific Field
        unsigned char bSrcAddr;      // Source Address
        unsigned int dwPGN;          // Parameter Group Number
        unsigned char bDLC;
        unsigned char bData[8];
} vcil_j1939_receive_message_t;
```

- **Description:**

This data structure defines the J1939 message.

- **Parameter**

dwID

The Identifier of CAN.

priority

The priority of this message. The priority of range is 0 to 7. Normally set to 6.

bDataPage

This bit expands the number of possible Parameter Groups that can be represented by the identifier.

bPDUFormat

The PDU format (PF) determines whether the message can be transmitted with a destination address

bPDUSpecific

If the PF is between 0 and 239, the message is addressable (PDU1) and the PS field contains the destination address.

If the PF is between 240 and 255, the message can only be broadcast

(PDU2) and the PS field contains a Group Extension.

bSrcAddr

The destination address of this message. 0x00 to 0xFF

dwPGN

The term Parameter Group Number (PGN) is used to refer to the value of the Reserve bit, DP, PF, and PS fields.

bDLC

The standard J1939 message data length.

bData

The data array of J1939 message.

2.11 apacer_obd2_message_ex_t

- **Syntax:**

```
typedef struct
{
    unsigned char bPort;
    unsigned int dwID;
    unsigned char bService;
    unsigned char bPID;
    unsigned char bMode;
    unsigned char bDLC;
    unsigned char bData[8];
} apacer_obd2_message_ex_t;
```

- **Description:**

This data structure defines the obd2 message.

- **Parameter**

bPort

This message come from/send to which port.

dwID

The Identifier of CAN.

bService

The OBD2 provide service id.

bPID

The OBD2 provide PID.

bMode

The type of CAN message. 0 is 11 bits. 1 is 29 bits..

bDLC

The standard OBD2 message data length.

bData

The data array of OBD2 message.

1. SDK Programming Common API

1.1 SusiCanbusInitialize

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusInitialize(void);

- **Description:**
  (TBD)
- **Parameters**
  None.

1.2 SusiCanbusGetCOMPort

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusGetCOMPort(uint32_t Id, char* auto_com);

- **Description:**
  This API only for Apacer CAN bus to auto gets COM port.
  RDC/VCIL CAN bus will not support auto get COM port API.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  auto_com[out]
  Got APACER serial COM port automatically, for example API returns integer '3' means "COM3".

1.3 SusiCanbusSetCapability

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusSetCapability(uint32_t Id, com_setting_t com_setting);

- **Description:**
  CAN port open, initialization and configuration.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

  com_setting_t[in][out]

  See com_setting_t

- The usage of com_setting_t for each SUSI ID

com_setting_t	IsEventMode	com1_num	com2_num
SUSI_ID_RDC_CAN0 SUSI_ID_RDC_CAN1	Not used	Not used	Not used
SUSI_ID_APACER_CAN0 SUSI_ID_APACER_CAN1	[in] 1: event mode 0: polling mode	[out] (auto get COM)	Not used
SUSI_ID_VCIL_CAN0 SUSI_ID_VCIL_CAN1 SUSI_ID_VCIL_CAN2	(TBD)	(TBD)	(TBD)

1.4 SusiCanbusSetRDCConfiguration

- **Syntax:**

```
SusiStatus_t SUSI_API SusiCanbusSetRDCConfiguration(uint32_t Id,
rdc_configuration_t rdc_config);
```

- **Description:**

  For RDC CAN port open, set configuration of baud rate and sampling rule.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

  rdc_configuration_t [in]

  See rdc_configuration_t

1.5 SusiGetFwVer

- **Syntax:**

SusiStatus_t SUSI_API SusiGetFwVer(uint32_t Id, char *version);

- **Description:**
  Get the version of CAN bus firmware.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  *version[out]
  The version number of character pointer.

1.6 SusiGetSDKVer

- **Syntax:**

SusiStatus_t SUSI_API SusiGetSDKVer(uint32_t Id, char *version);

- **Description:**
  Get the version of CAN bus SDK.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  *version[out]
  The version number of character pointer.

1.7 SusiGetVendor

- **Syntax:**

SusiStatus_t SUSI_API SusiGetFwVer(uint32_t Id, char *version);

- **Description:**
  Get the CAN bus hardware vendor name.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

  *vendor[out]

  The hardware vendor name of character pointer.

1.8 SusiCanbusRead

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusRead(uint32_t Id, vcil_can_message_t *message);

- **Description:**

  Get the version of CAN bus firmware.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

  vcil_can_message_t[out]

  Pointer to vcil_can_message_t which is used to store received CAN message

1.9 SusiCanbusWrite

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusWrite(uint32_t Id, vcil_can_message_t *message);

- **Description:**

  Write a CAN message to specified CAN port.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

  vcil_can_message_t[in]

  Pointer to vcil_can_message_t which stores the CAN message to be sent

1.10 SusiCanbusClearBuffer

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusClearBuffer (uint32_t Id);

- **Description:**
  This API will clear TX and RX bus data.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

1.11 SusiCanbusUninitialize

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusUninitialize (void);

- **Description:**
  TX/RX bus close, release handle, resource and un-initialization.

- **Parameters**
  None.

2. SDK Programming Setting API

2.1 SusiCanbusSetSpeed

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusSetSpeed(uint32_t Id, vcil_can_speed speed);

- **Description:**

  Set the specified CAN port bus baud rate.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

  vcil_can_speed[in]

  The bus baud rate, see vcil_can_speed

2.2 SusiCanbusGetSpeed

- **Syntax:**

SusiStatus_t SUSI_API SusiCanbusGetSpeed(uint32_t Id, vcil_can_speed *speed);

- **Description:**

  Get the specified CAN port bus baud rate.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

  *vcil_can_speed[in]

  The bus baud rate, see vcil_can_speed

3. SDK Programming Filter API

3.1 Support list

CAN bus hardware	Support Filter API
RDC	Not Support
APACER	Support
VCIL	Support

3.2 SusiCanbusSetMask

SusiStatus_t SUSI_API SusiCanbusSetMask(uint32_t Id, vcil_can_mask_t *mask);

- **Description:**
  Set the specified CAN message filter to specified filter bank of specified CAN port and enable it.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  vcil_can_mask_t[in]
  The mask configuration. See vcil_can_mask_t

3.3 SusiCanbusGetMask

SusiStatus_t SUSI_API SusiCanbusGetMask(uint32_t Id, vcil_can_mask_t *mask);

- **Description:**
  Get the CAN message filter from specified filter bank of specified CAN port.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  vcil_can_mask_t[in]

The mask configuration. See vcil_can_mask_t

3.4 SusiCanbusRemoveMask

SusiStatus_t SUSI_API SusiCanbusRemoveMask(uint32_t Id, unsigned char bank);

- **Description:**
  Remove a filter from specified filter bank of specified CAN port.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  bank [in]
  The bank of the mask to be removed.

3.5 SusiCanbusResetMask

SusiStatus_t SUSI_API SusiCanbusResetMask(uint32_t Id);

- **Description:**
  Reset all filter bank of the specified CAN port.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

4. SDK Programming Others API

4.1 SusiCanbusSetEvent

SusiStatus_t SUSI_API SusiCanbusSetEvent(uint32_t Id, void *can_rx_event);

- **Description:**
  Only VCIL CAN bus use this API.
  Set a user define event in order to let VCIL library notify the specified event when CAN message is received.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  can_rx_event[in]
  Pointer to the CAN received event. In windows, the can_rx_event will pointer to a Windows Events HANDLE.

4.2 SusiCanbusGetErrorStatus

SusiStatus_t SUSI_API SusiCanbusGetErrorStatus(uint32_t Id, vcil_can_error_status *status);

- **Description:**
  Get the specified CAN port error status. this API can be using to detect bus error status.
- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  *status[in]
  See vcil_can_error_status

5. J1939 SDK Programming API

5.1 SusiCanbusJ1939Read

SusiStatus_t SUSI_API SusiCanbusJ1939Read(uint32_t Id, vcil_j1939_receive_message_t *message);

- **Description:**
  Get a J1939 message from library J1939 buffer if available otherwise you may get an invalid J1939 message.
- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  vcil_j1939_receive_message_t [out]
  Pointer to vcil_j1939_receive_message_tvcil_j1939_receive_message_t which is used to store received CAN message

5.2 SusiCanbusJ1939Write

SusiStatus_t SUSI_API SusiCanbusJ1939Write (uint32_t Id, vcil_j1939_message_t *message);

- **Description:**
  Write a J1939 message from library J1939 buffer.
- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  vcil_j1939_message_t [out]
  Pointer to vcil_j1939_message_t structure that store the J1939 message to be sent.

5.3 SusiCanbusJ1939AddMask

SusiStatus_t SUSI_API SusiCanbusJ1939AddMask(uint32_t Id, unsigned int pgn);

- **Description:**
  Add a PGN mask to specify CAN port. The mask will allow message with specified PGN able to pass through the filter and be received by user APP.
  .
- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  pgn [in]
  The PGN mask.

5.4 SusiCanbusJ1939GetAllMask

SusiStatus_t SUSI_API SusiCanbusJ1939GetMaskNumber(uint32_t Id, unsigned int* total);

- **Description:**
  Get all the number of J1939 mask of specified CAN port.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  *total [out]
  The total number of mask PGN.

5.5 SusiCanbusJ1939RemoveMask

SusiStatus_t SUSI_API SusiCanbusJ1939RemoveMask(uint32_t Id, unsigned int pgn);

- **Description:**

Remove specified PGN mask from the specified CAN port.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

  pgn[in]

  The PGN number would be removed.

5.6 SusiCanbusJ1939RemoveAllMask

```
SusiStatus_t SUSI_API SusiCanbusJ1939RemoveAllMask(uint32_t Id);
```

- **Description:**

  Remove all J1939 mask from the specified CAN port.

- **Parameters**

  Id[in]

  See SUSI CAN bus ID

6. OBD2 SDK Programming API

6.1 SusiCanbusOBD2WriteEx

SusiStatus_t SUSI_API SusiCanbusOBD2WriteEx(uint32_t Id, apacer_obd2_message_ex_t *message);

- **Description:**
  Write OBD2 message to specify CAN port.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  *apacer_obd2_message_ex_t[in]
  Pointer to apacer_obd2_message_ex_t structure that store the J1939 message to be sent.

6.2 SusiCanbusOBD2ReadEx

SusiStatus_t SUSI_API SusiCanbusOBD2ReadEx(uint32_t Id, apacer_obd2_message_ex_t *message);

- **Description:**
  Get an OBD2 message from VCIL library OBD2 buffer if available otherwise you may get an invalid OBD2 message.

- **Parameters**
  Id[in]
  See SUSI CAN bus ID

  *apacer_obd2_message_ex_t[out]
  Pointer to apacer_obd2_message_ex_t structure which is used to store received OBD2 message