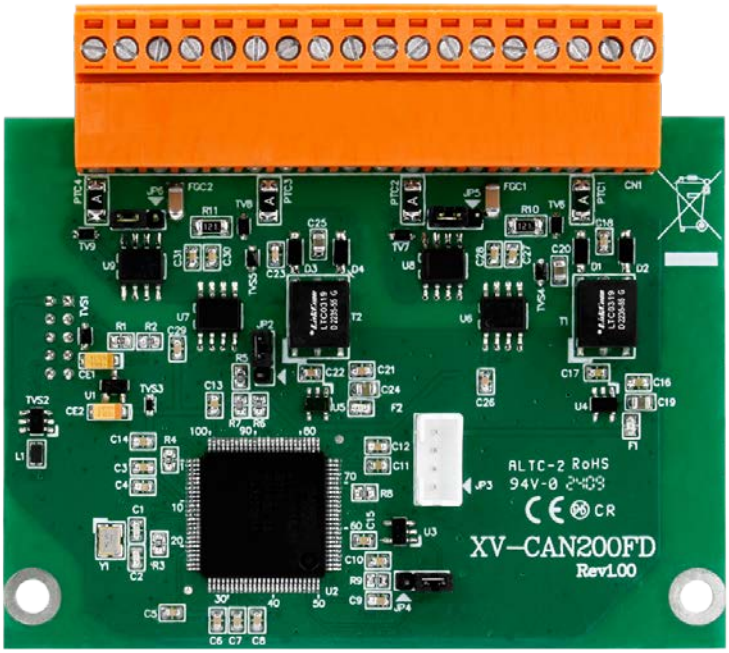
XV-CAN200FD
User Manual

Version 1.0.1, Feb. 2026



Service and usage information for

XV-CAN200FD

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product.ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Contact us

If you have any problem, please feel free to contact us.
You can count on us for quick response.

Email: service@icpdas.com

Table of Contents

1. Introduction

The XV-CAN200FD provides 2-port isolation CAN FD expansion for LP-2241M, LP-2841M. Each channel features Isolation CAN port. It provides quick and easy expansion of CAN Bus communication interfaces for LP-2241M, LP-2841M applications.

1.1. Specifications

Model Name	XV-CAN200FD
CAN Interface	
Channel Number	2
Connector	18-pin terminal-block connector
Transmission Speed	CAN bit rates: 10 ~ 1000 kbps, CAN FD bit rates for data field: 100 ~ 10000 kbps
Terminal Resistor	Jumper switch for the 120 Ω terminal resistor
Isolation	3000 VDC for DC-to-DC, 2500 Vrms for photo-coupler
Specification	ISO 11898-2, CAN 2.0 A/B and FD
Power	
Power Consumption	1.5 W (Max.)
Mechanism	
Dimensions	59 mm x 82 mm x 13 mm (W x L x H)
Environment	
Operating Temp.	-25 ~ 75 ℃
Storage Temp.	-30 ~ 80 ℃
Humidity	10 ~ 90% RH, non-condensing

1.2. Features

- Compatible with the ISO 11898-2 standard
- Compatible with CAN specification 2.0 A/B and FD
- CAN FD support for ISO and Non-ISO (Bosch) standards switchable
- CAN FD bit rates for data field from 100 kbps to 10000 kbps
- CAN bit rates from 10 kbps to 1000 kbps
- Support CAN Bus message filter configuration
- CAN message timestamp resolution 1ms.
- Built-in jumper-switch to select 120 ohm terminal resister for CAN Bus

2. Technical data

2.1. Block Diagram

The following figure is the block diagram illustrating the functions of the XV-CAN200FD.

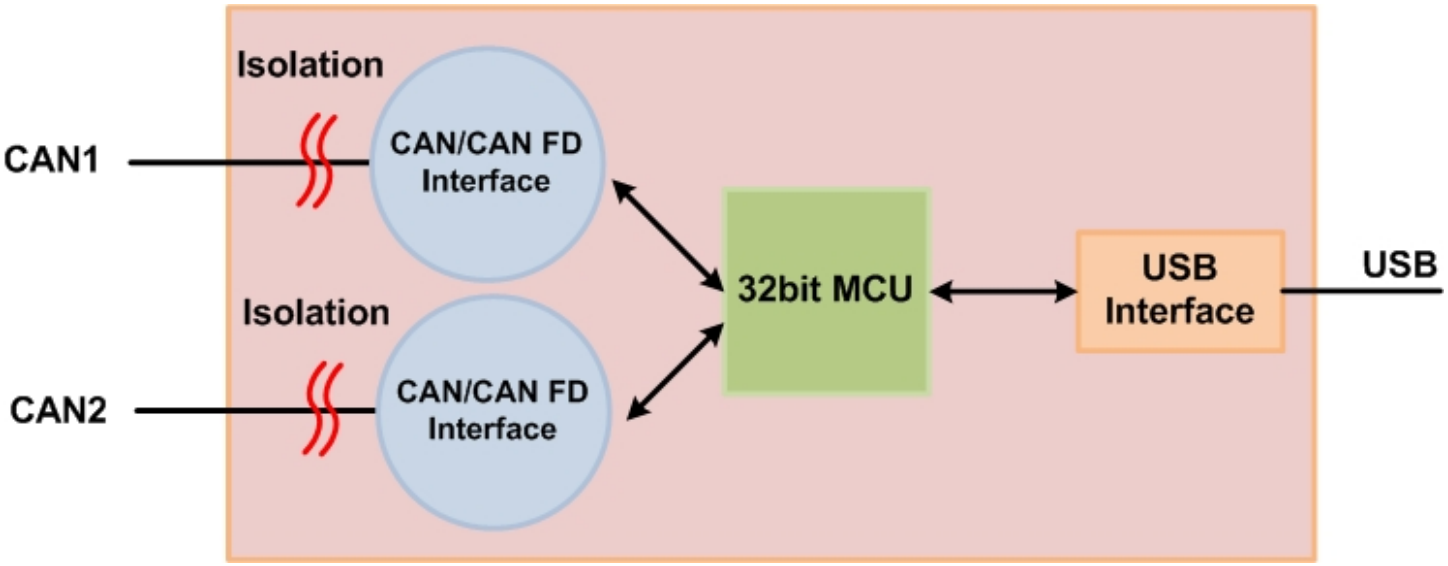
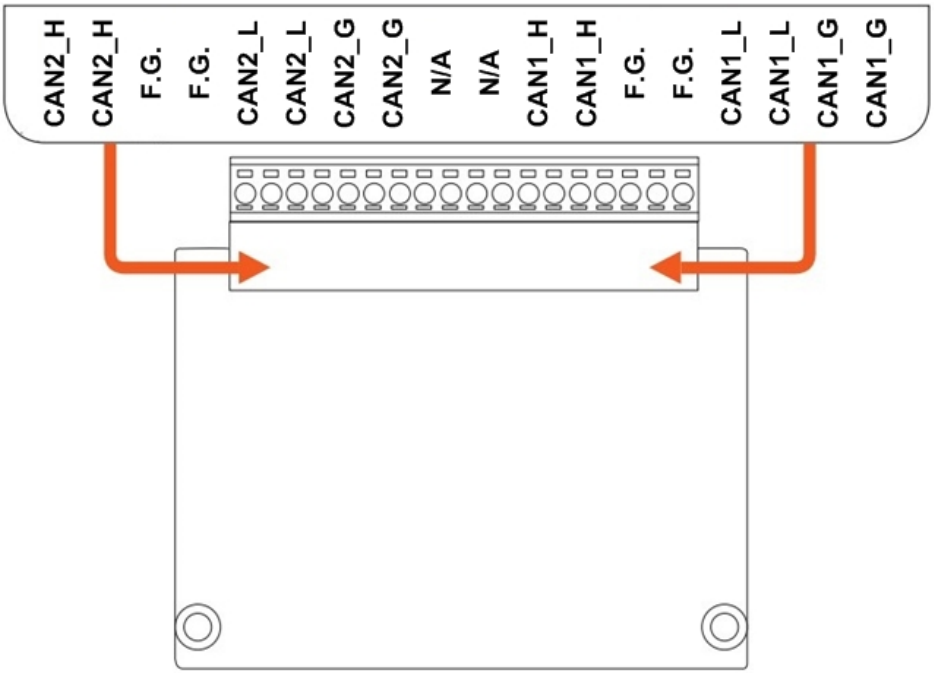


Figure 2-1-1 Block Diagram of XV-CAN200FD

2.2. Pin Assignment



The pin assignments of 18-pin terminal block connector of XV-CAN200FD is shown in the below tables.

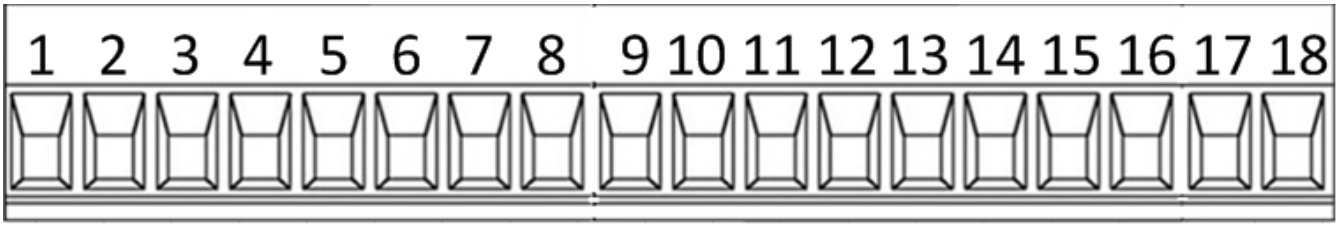


Table 2-2-1 Pin Assignment

Pin No	Name	Description
1, 2	CAN1_G	CAN ground of CAN1 port
3, 4	CAN1_L	CAN_Low bus line of CAN1 port.
5, 6	F.G.	Frame Ground.
7, 8	CAN1_H	CAN_High bus line of CAN1 port.
9, 10	N/A	N/A
11, 12	CAN2_G	CAN ground of CAN2 port
13, 14	CAN2_L	CAN_Low bus line of CAN2 port.
15, 16	F.G.	Frame Ground.
17,18	CAN2_H	CAN_High bus line of CAN2 port.

2.3. Terminal Resistor Setup

In order to minimize the reflection effects on the CAN Bus line, the CAN Bus line has to be terminated at both ends by two terminal resistors as in the following figure. According to the ISO 11898-2 specification, each terminal resistor is 120Ω (or between 108Ω~132Ω). The bus topology and the positions of these terminal resistors are shown as following figure.

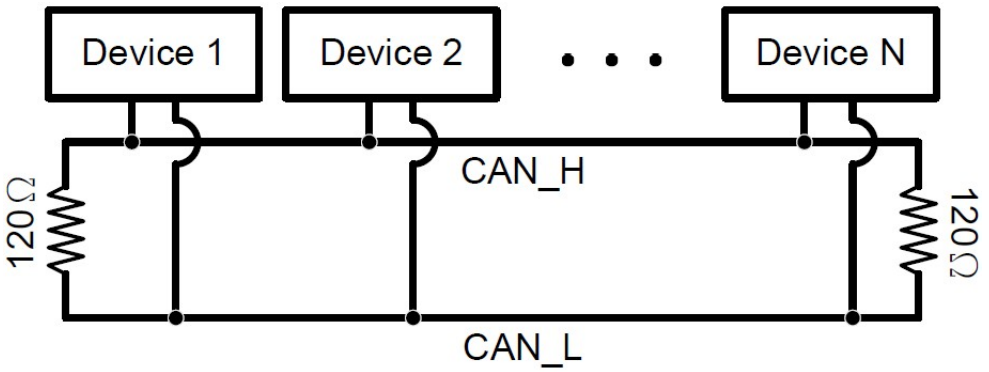


Figure 2-3-1 CAN Bus network topology

Each XV-CAN200FD includes one build-in 120Ω terminal resistors for each CAN port, users can decide if it is enabled or not. The jumper switches for terminal resistor are JP5 and JP6.

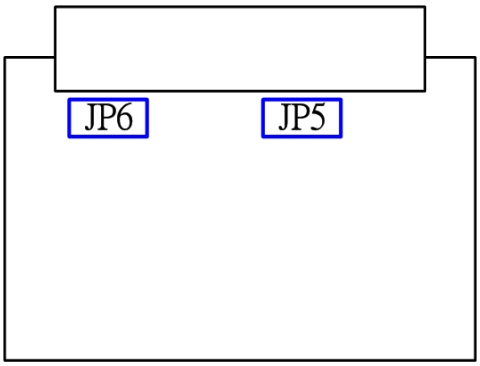
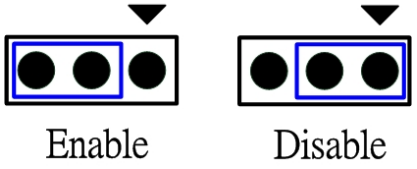


Figure 2-3-2 Location of Terminal Resistor Jumper Switch of XV-CAN200FD

The following jumper switch statuses present the condition if the terminal resistor is active (default) or inactive.

Table 2-3-1 Adjustment of Terminal Resistor

JP5/JP6	Pin No.	Description
	*JP5*	*Enable: Active CAN1 terminal resistor (default)* *Disable: Inactive CAN1 terminal resistor*
	JP6	*Enable: Active CAN2 terminal resistor (default)* *Disable: Inactive CAN2 terminal resistor*

2.4. Wire Connection

The wire connection of the XV-CAN200FD is displayed below.

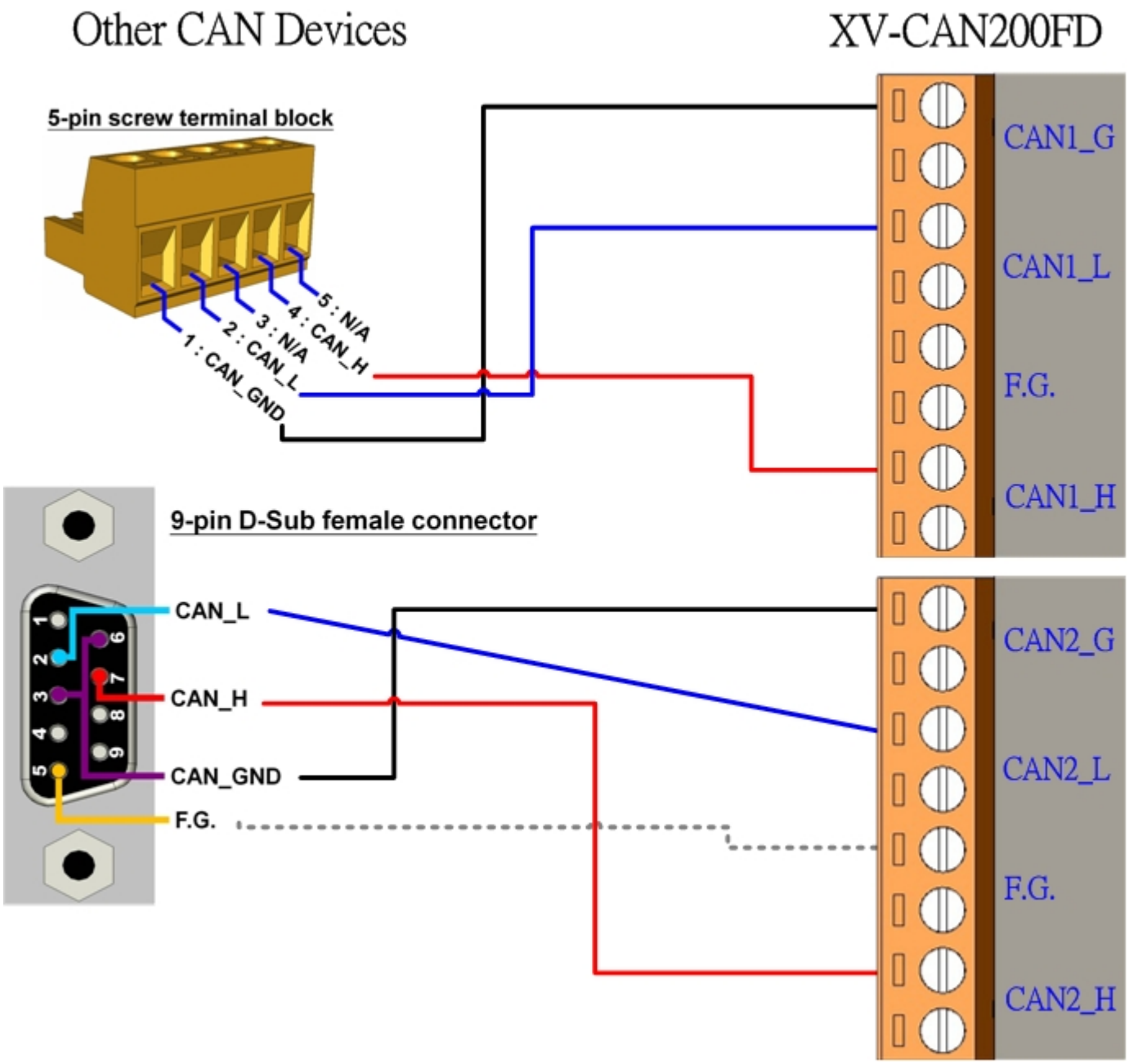


Figure 2-4-1 Wire Connection for XV-CAN200FD

3. Network Deployment

3.1. Driving Capability

Before introducing the driving capability (DC) of the XV-CAN200FD, some characteristics of copper cable must be assumed. The DC parameter follows the table show below.

Table 3-1-1 Recommended DC parameter for CAN Bus Line

Wire Cross-Section [mm^2]	Resistance [Ω/km]
~0.25 (AWG23)	< 90
~0.5 (AWG20)	< 50
~0.8 (AWG18)	< 33
~1.3 (AWG16)	< 20

Under the condition described above, users can refer to the following table to know the maximum node number in each segment following ISO 11898-2 and the maximum segment length when using different type of wire.

Table 3-1-2 Driving Capability

Wire Cross-Section [mm^2]	The maximum segment length [m] under the case of specific node number in this segment			
	16 Nodes	32 Nodes	64 Nodes	100 Nodes
~0.25 (AWG23)	< 220	< 200	< 170	< 150
~0.5 (AWG20)	< 390	< 360	< 310	< 270
~0.8 (AWG18)	< 590	< 550	< 470	< 410
~1.3 (AWG16)	< 980	< 900	< 780	< 670

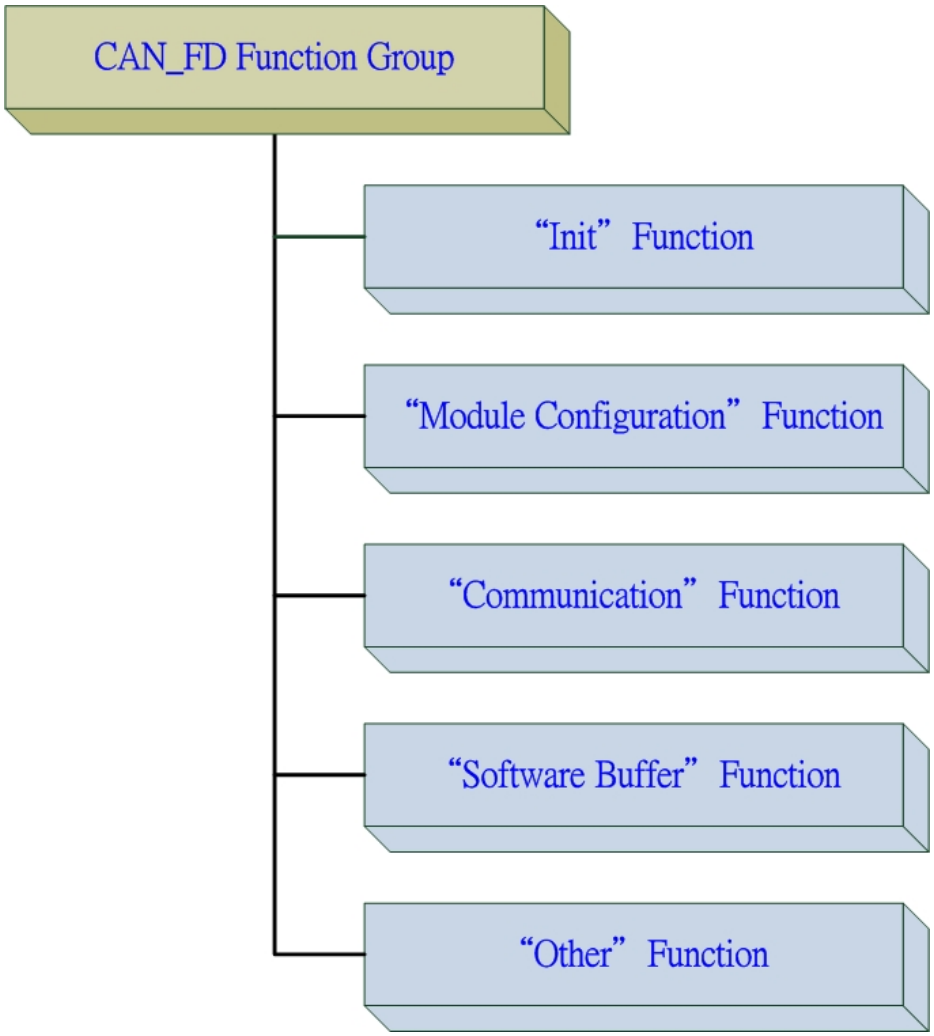
4. API Library

Users can develop own CAN Bus program by XV-CAN200FD API library quickly and easily. The CAN_FD library and demos can be downloaded from following web site.

The library and demos for LP-2241M, LP-2841M are located at:
https://www.icpdas.com/en/download/show.php?num=9583

4.1. API Library Overview

All the functions provided by CAN_FD library can be separated into five groups shown in following picture.



[Init Function]

These functions are used to scan and open/close the valid and necessary XV-CAN200FD device.

[Module Configuration Function]

These functions are used to set/get the parameters or information of XV-CAN200FD.

[Communication Function]

These functions are used to send/receive CAN/CAN FD messages through XV-CAN200FD

[Software Function]

All the transmitted/received CAN/CAN FD messages will be saved in software buffer provided by CAN_FD library first. These related software functions are used to operate the software buffer of CAN_FD library.

[Other Function]

These functions are used to get the CAN_FD library information or helpful for users program.

4.2. API Library Function Table

All the API functions provided in the CAN_FD library are listed in the following table.

"Init" Function Table		
No.	Function Name	Description
1	CANFD_ScanDevice	Scan all the valid device from the PAC
2	CANFD_ListDevice	List all the valid devices to a pid/bid table
3	CANFD_OpenDevice	Open a necessary device via pid/bid setting.
4	CANFD_CloseDevice	Close a selected device.

"Module Configuration" Function Table		
No.	Function Name	Description
1	CANFD_SetCANOPMode	Set the enable/bus monitoring/ISO mode in the assigned CAN port
2	CANFD_GetCANOPMode	Get the enable/bus monitoring/ISO mode in the assigned CAN port
3	CANFD_SetCANADBaudRate	Set the arbitration/data phase bit rate in the assigned CAN port
4	CANFD_GetCANADBaudRate	Get the arbitration/data phase bit rate in the assigned CAN port
5	CANFD_SetCANGlobalFilter	Set the CAN filter setting of remote frame in the assigned CAN port
6	CANFD_GetCANGlobalFilter	Get the CAN filter setting of remote frame and standard/extended ID list sizes in the assigned CAN port
7	CANFD_SetCANSTDIDFilter	Set the CAN filter setting of standard IDs and standard ID list sizes in the assigned CAN port
8	CANFD_GetCANSTDIDFilter	Get the CAN filter setting of standard IDs in the assigned CAN port
9	CANFD_SetCANEXTIDFilter	Set the CAN filter setting of extended IDs and extended ID list sizes in the assigned CAN port
10	CANFD_GetCANEXTIDFilter	Get the CAN filter setting of extended IDs in the assigned CAN port
11	CANFD_GetCANStatus	Get the CAN Bus status in the assigned CAN port
12	CANFD_SetCANReInitialize	Restarts the CAN port on the specified CAN port.

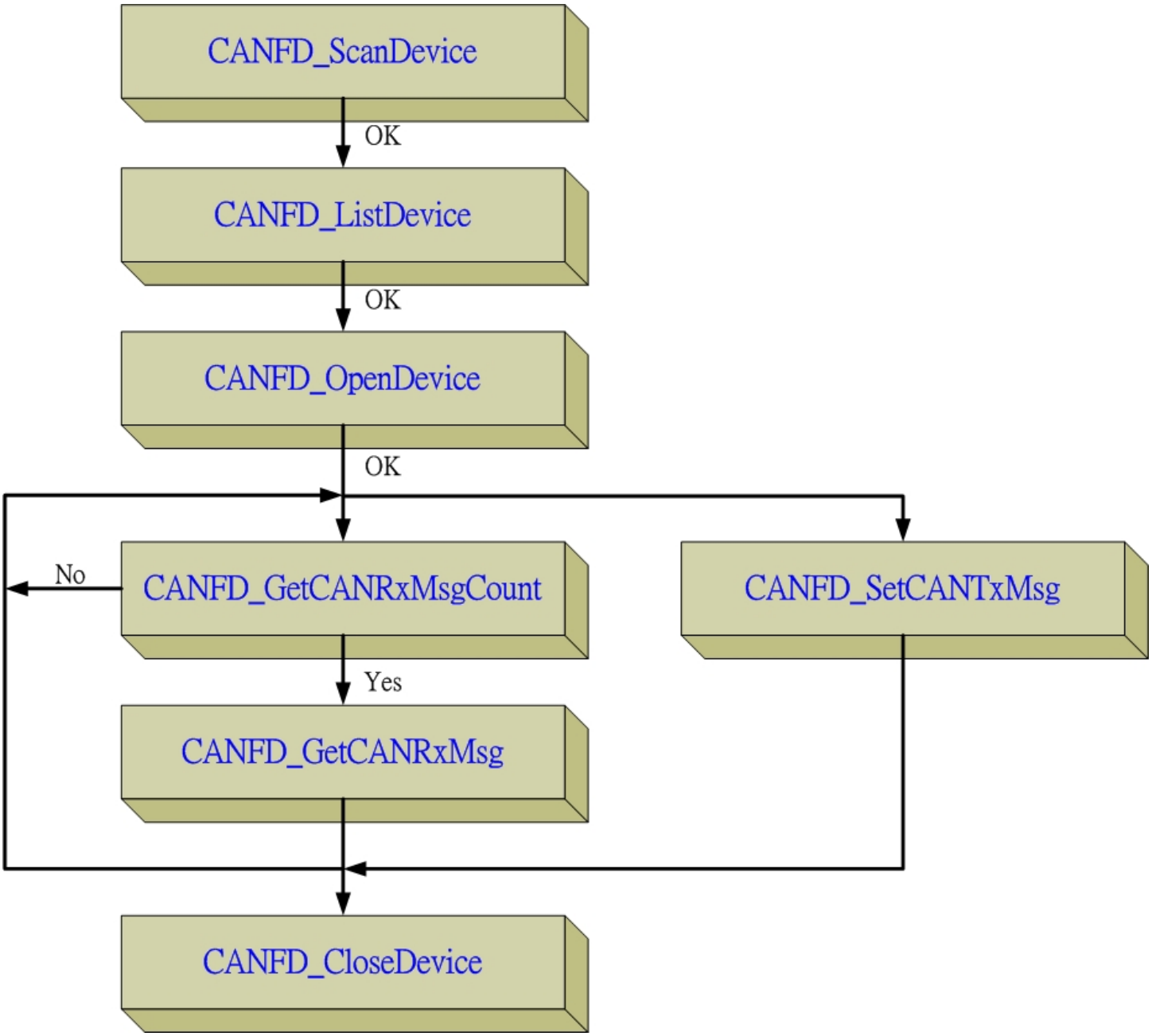
"Communication" Function Table		
No.	Function Name	Description
1	CANFD_SetCANTxMsg	Send a CAN/CAN FD message to the software transmitted buffer of the assigned CAN port
2	CANFD_GetCANRxMsg	Get a CAN/CAN FD message from the software received buffer of the assigned CAN port
3	CANFD_SetCANHWSendMode	Enable/disable the hardware cyclic sending CAN/CAN FD message mode in the assigned CAN port.
4	CANFD_GetCANHWSendMode	Get the hardware cyclic sending CAN/CAN FD message mode in the assigned CAN port
5	CANFD_SetCANHWSendMsg	Set the hardware cyclic sending CAN/CAN FD message content in the assigned CAN port.
6	CANFD_GetCANRxFramePerSec	Get the CAN Bus data flow in the assigned CAN port

"Software Buffer" Function Table		
No.	Function Name	Description
1	CANFD_GetCANRxMsgCount	Get the received CAN/CAN FD message counts in the software buffer of the assigned CAN port
2	CANFD_ClearCANRxBuf	Clear the CAN/CAN FD message in the software received buffer of the assigned CAN port
3	CANFD_ClearCANTxBuf	Clear the CAN/CAN FD message in the software transmitted buffer of the assigned CAN port

"Other" Function Table		
No.	Function Name	Description
1	CANFD_GetDllVersion	Get the API library version.
2	CANFD_GetFwVer	Get the firmware version of the module
3	CANFD_SetSN	Set the module's BID (board ID)
4	CANFD_ResetModule	Reset module

4.3. API Library Flow Diagram

The following is the basic control flow chart of user's CAN Bus program development by using CAN_FD API Library shown in following picture.



4.4. Init Functions

These functions are used to scan and open/close the valid and necessary XV-CAN200FD device.

4.4.1 CANFD_ScanDevice

This function is used to scan all the valid XV-CAN200FD devices on PC side.

Syntax:

C++
int CANFD_ScanDevice(void);
C#
Int32 CANFD_ScanDevice();

Parameter:

None.

Return Value:

Return 0 means success, others means failure.

4.4.2 CANFD_ListDevice

The API library maximum support eight XV-CAN200FD devices in the same PAC. This function is used to list all the scanned XV-CAN200FD devices' PID (product ID) and BID (board ID).

Syntax:

C++
BYTE CANFD_ListDevice(WORD* o_wPID, DWORD* o_dwBID);
C#
Byte CANFD_ListDevice(UInt16[] o_wPID, Uint32[] o_dwBID);

Parameter:

o_wPID

[out] The pointer is used to receive maximum eight PID (product ID) of XV-CAN200FD devices

o_dwBID

[out] The pointer is used to receive maximum eight BID (board ID) of XV-CAN200FD devices'

Return Value:

Return the amount of valid XV-CAN200FD devices that API library scanned.

4.4.3 CANFD_OpenDevice

This function is used to open the necessary XV-CAN200FD device. After using the pid and bid to open the device, users can get a device ID and can use this ID with "Communication" functions to send/receive CAN messages.

Syntax:

C++

```
int CANFD_OpenDevice(WORD *o_wDevice_id, WORD i_wpid, DWORD i_wbid);
```

C#

```
Int32 CANFD_OpenDevice(out Uint16 o_wDevice_id, Uint16 i_wpid, Uint32 i_wbid);
```

Parameter:

***o_wDevice_id**

[out] The pointer is used to receive a necessary device ID in the assigned pid and bid of XV-CAN200FD device.

I_wpid

[in] The PID (product ID) of the XV-CAN200FD device which needs to be open.

I_wbid

[in] The BID (board ID) of the XV-CAN200FD device which needs to be open.

Return Value:

Return 0 means success, others means failure.

4.4.4 CANFD_CloseDevice

This function is used to close the XV-CAN200FD device. After the device closed, all the resources the API Library used will be released.

Syntax:

C++
int CANFD_CloseDevice(WORD i_wDevice_id);
C#
Int32 CANFD_CloseDevice(Uint16 i_wDevice_id);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

Return Value:

Return 0 means success, others means failure.

4.5. Module Configuration Functions

These functions are used to set/get the parameters or information of XV-CAN200FD.

4.5.1 CANFD_SetCANOPMode

This function is used to enable/disable, set normal or bus monitoring mode and CAN FD ISO/Non-ISO mode in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_SetCANOPMode(WORD i_wDevice_id, BYTE i_byCANPort, WORD i_wEnable, WORD i_wBusMode, WORD i_wISOMode);
C#
Int32 CANFD_SetCANOPMode(Uint16 i_wDevice_id, Byte i_byCANPort, Uint16 i_wEnable, Uint16 i_wBusMode, Uint16 i_wISOMode);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

I_wEnable

[in] Enable or disable the assigned CAN port of the XV-CAN200FD device.
0: disable, 1: enable.

I_wBusMode

[in] Set normal or bus monitoring mode of the assigned CAN port of the XV-CAN200FD device.
0: normal mode, 1: bus monitoring mode.

I_wISOMode

[in] Set CAN FD ISO or Non-ISO mode of the assigned CAN port of the XV-CAN200FD device.
0: ISO mode, 1: Non-ISO mode.

Return Value:

Return 0 means success, others means failure.

4.5.2 CANFD_GetCANOPMode

This function is used to get the enable/disable setting, normal or bus monitoring mode and CAN FD ISO/Non-ISO mode in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++

```
int CANFD_GetCANOPMode(WORD i_wDevice_id, BYTE i_byCANPort, WORD* o_wEnable, WORD* o_wBusMode, WORD* o_wISOMode);
```

C#

```
Int32 CANFD_GetCANOPMode(Uint16 i_wDevice_id, Byte i_byCANPort, out Uint16 o_wEnable, out Uint16 o_wBusMode, out Uint16 o_wISOMode);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_wEnable***

[out] Enable or disable the assigned CAN port of the XV-CAN200FD device.
0: disable, 1: enable.

****o_wBusMode***

[out] Normal or bus monitoring mode of the assigned CAN port of the XV-CAN200FD device.
0: normal mode, 1: bus monitoring mode.

****o_wISOMode***

[out] CAN FD ISO or Non-ISO mode of the assigned CAN port of the XV-CAN200FD device.
0: ISO mode, 1: Non-ISO mode.

Return Value:

Return 0 means success, others means failure.

4.5.3 CANFD_SetCANADBaudRate

This function is used to set the operating CAN/CAN FD arbitration and data phase bit rate in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_SetCANADBaudRate(WORD i_wDevice_id, BYTE i_byCANPort, DWORD i_dwArbitrBR, DWORD i_dwDataBR, WORD i_wArbitrBRSP, WORD i_wDataBRSP);
C#
Int32 CANFD_SetCANADBaudRate(Uint16 i_wDevice_id, Byte i_byCANPort, Uint32 i_dwArbitrBR, Uint32 i_dwDataBR, Uint16 i_wArbitrBRSP, Uint16 i_wDataBRSP);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

I_dwArbitrBR

[in] The bit rate configured for the CAN and CAN FD arbitration phase in the assigned CAN port of the XV-CAN200FD device. Unit: bps (bit per second).
Valid Range: 10000 ~ 1000000 (10 kbps ~ 1000 kbps).

I_dwDataBR

[in] The bit rate configured for the CAN FD data phase in the assigned CAN port of the XV-CAN200FD device. Unit: bps (bit per second).
Valid Range: 100000 ~ 10000000 (100 kbps ~ 10000 kbps).
Remark:
The bit rate configured for CAN FD data phase (***i_dwDataBR***) must be higher or equal to the bit rate configured for the arbitration phase (***i_dwArbitrBR***).

I_wArbitrBRSP

[in] The sample point of CAN bit rate and CAN FD arbitration phase bit rate in

the assigned CAN port of the XV-CAN200FD device. Unit: 0.01%, 8750 means 87.50%.
Suggested Range: 7500 ~ 8750 (75.00% ~ 87.50%).

I_wDataBRSP

[in] The sample point of CAN FD data phase bit rate in the assigned CAN port of the XV-CAN200FD device. Unit: 0.01%, 8750 means 87.50%.
Suggested Range: 7500 ~ 8750 (75.00% ~ 87.50%).

Return Value:

Return 0 means success, others means failure.

4.5.4 CANFD_GetCANADBaudRate

This function is used to get the current CAN/CAN FD arbitration and data phase bit rate in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_GetCANADBaudRate(WORD i_wDevice_id, BYTE i_byCANPort, DWORD* o_dwArbitrBR, DWORD* o_dwDataBR, WORD* o_wArbitrBRSP, WORD* o_wDataBRSP); **C#**
Int32 CANFD_GetCANADBaudRate(Uint16 i_wDevice_id, Byte i_byCANPort, out Uint32 o_dwArbitrBR, out Uint32 o_dwDataBR, out Uint16 o_wArbitrBRSP, out Uint16 o_wDataBRSP);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_dwArbitrBR***

[out] The CAN bit rate and CAN FD arbitration phase bit rate in the assigned CAN port of the XV-CAN200FD device. Unit: bps (bit per second).
Valid Range: 10000 ~ 1000000 (10 kbps ~ 1000 kbps).

****o_dwDataBR***

[out] The CAN FD data phase bit rate in the assigned CAN port of the XV-CAN200FD device. Unit: bps (bit per second).
Valid Range: 100000 ~ 10000000 (100 kbps ~ 10000 kbps).

****o_wArbitrBRSP***

[out] The sample point of CAN bit rate and CAN FD arbitration phase bit rate in

the assigned CAN port of the XV-CAN200FD device. Unit: 0.01%, 8750 means 87.50%.

***o_wDataBRSP**

[out] The sample point of CAN FD data phase bit rate in the assigned CAN port Unit: 0.01%, 8750 means 87.50%.

Return Value:

Return 0 means success, others means failure.

4.5.5 CANFD_SetCANGlobalFilter

This function is used to set CAN filter function of “reject remote standard or extended CAN ID frame” in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++

```
int CANFD_SetCANGlobalFilter(WORD i_wDevice_id, BYTE i_byCANPort, BYTE
i_byRejectRFS, BYTE i_byRejectRFE);
```

C#

```
Int32 CANFD_SetCANGlobalFilter(Uint16 i_wDevice_id, Byte i_byCANPort, Byte
i_byRejectRFS, Byte i_byRejectRFE);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

I_byRejectRFS

[in] Reject the remote standard CAN ID frame in the assigned CAN port of the XV-CAN200FD device.
0: pass the remote standard CAN ID frame
1: reject the remote standard CAN ID frame

i_byRejectRFE

[in] Reject the remote extended CAN ID frame in the assigned CAN port of the XV-CAN200FD device.
0: pass the remote extended CAN ID frame
1: reject the remote extended CAN ID frame

Return Value:

Return 0 means success, others means failure.

4.5.6 CANFD_GetCANGlobalFilter

This function is used to get CAN filter function of “reject remote standard or extended CAN ID frame” and “standard and extended CAN/CAN FD ID lists size” parameters in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++

```
int CANFD_GetCANGlobalFilter(WORD i_wDevice_id, BYTE i_byCANPort, BYTE *o_byRejectRFS, BYTE *o_byRejectRFE, WORD* o_wSTDFIDListSize, WORD* o_wEXTFIDListSize);
```

C#

```
Int32 CANFD_GetCANGlobalFilter(Uint16 i_wDevice_id, Byte i_byCANPort, out Byte o_byRejectRFS, out Byte o_byRejectRFE, out Uint16 o_wSTDFIDListSize, out Uint16 o_wEXTFIDListSize);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_byRejectRFS***

[out] *The* “Reject the remote standard CAN ID frame” parameter in the assigned CAN port of the XV-CAN200FD device.
0: pass the remote standard CAN ID frame
1: reject the remote standard CAN ID frame

****o_byRejectRFE***

[out] *The* “Reject the remote extended CAN ID frame” parameter in the assigned CAN port of the XV-CAN200FD device.
0: pass the remote extended CAN ID frame
1: reject the remote extended CAN ID frame

o_wSTDFIDListSize

[out] The CAN filter function of “standard CAN ID list size” parameter in the assigned CAN port of the XV-CAN200FD device. This parameter is used for valid CAN ID filter ranges on **CANFD_GetCANSTDIDFilter** API function. Valid Range: 0 ~ 128.

o_wEXTFIDListSize

[out] The CAN filter function of “extended CAN ID list size” parameter in the assigned CAN port of the XV-CAN200FD device. This parameter is used for valid CAN ID filter ranges on **CANFD_GetCANEXTIDFilter** API function. Valid Range: 0 ~ 64.

Return Value:

Return 0 means success, others means failure.

4.5.7 CANFD_SetCANSTDIDFilter

This function is used to set CAN filter function of “standard CAN/CAN FD ID list size” and “standard CAN/CAN FD ID ranges” parameters in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++

```
int CANFD_SetCANSTDIDFilter(WORD i_wDevice_id, BYTE i_byCANPort, WORD
i_wSTDFIDListSize, WORD* i_wSTDFID1, WORD* i_wSTDFID2);
```

C#

```
Int32 CANFD_SetCANSTDIDFilter(Uint16 i_wDevice_id, Byte i_byCANPort, Uint16
i_wSTDFIDListSize, [In, Out] Uint16[] i_wSTDFID1, [In, Out] Uint16[] i_wSTDFID2);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

I_wSTDFIDListSize

[in] The CAN filter function of “standard CAN ID list size” parameter in the assigned CAN port of the XV-CAN200FD device. Maximum support 128 CAN standard ID filter in each CAN port.

****i_wSTDFID1, *i_wSTDFID2***

[in/out] This point to arrays of the elements of standard CAN/CAN FD ID filter ranges in the assigned CAN port of the XV-CAN200FD device. Valid used size of the arrays are depended by the “***i_wSTDFIDListSize***” parameter. The contents of each element are defined as below:
Element0: CAN ID from i_wSTDFID1[0] to i_wSTDFID2[0]
Element1: CAN ID from i_wSTDFID1[1] to i_wSTDFID2[1]
…
Element127: CAN ID from i_wSTDFID1[127] to i_wSTDFID2[127]

Return Value:

Return 0 means success, others means failure.

4.5.8 CANFD_GetCANSTDIDFilter

This function is used to get CAN filter function of “standard CAN/CAN FD ID ranges” parameter in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++

```
int CANFD_GetCANSTDIDFilter(WORD i_wDevice_id, BYTE i_byCANPort, WORD* o_wSTDFID1, WORD* o_wSTDFID2);
```

C#

```
Int32 CANFD_GetCANSTDIDFilter(Uint16 i_wDevice_id, Byte i_byCANPort, [In, Out] Uint16[] o_wSTDFID1, [In, Out] Uint16[] o_wSTDFID2);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

o_wSTDFID1, *o_wSTDFID2

[in/out] This point to arrays of the elements of standard CAN ID filter ranges in the assigned CAN port of the XV-CAN200FD device. Each array must reserve space for saving maximum 128 elements of CAN/CAN FD standard ID. Valid used size of the arrays are depended by the “**o_wSTDFIDListSize**” parameter in the **CANFD_GetCANGlobalFilter()** API function. The contents of each element are defined as below:
Element0: CAN ID from i_wSTDFID1[0] to i_wSTDFID2[0]
Element1: CAN ID from i_wSTDFID1[1] to i_wSTDFID2[1]
…
Element127: CAN ID from i_wSTDFID1[127] to i_wSTDFID2[127]

Return Value:

Return 0 means success, others means failure.

4.5.9 CANFD_SetCANEXTIDFilter

This function is used to set CAN filter function of “extended CAN ID list size” and “extended CAN ID ranges” parameters in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++

```
int CANFD_SetCANEXTIDFilter(WORD i_wDevice_id, BYTE i_byCANPort, WORD
i_wEXTFIDListSize, DWORD* i_dwEXTFID1, DWORD* i_dwEXTFID2);
```

C#

```
Int32 CANFD_SetCANEXTIDFilter(Uint16 i_wDevice_id, Byte i_byCANPort, Uint16
i_wEXTFIDListSize, [In, Out] Uint16[] i_wEXTFID1, [In, Out] Uint16[] i_wEXTFID2);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

I_wEXTFIDListSize

[in] The CAN filter function of “extended CAN ID list size” parameter in the assigned CAN port of the XV-CAN200FD device. Maximum support 64 CAN extended ID filter in each CAN port.

****i_wEXTFID1, *i_wEXTFID2***

[in/out] This point to arrays of the elements of extended CAN ID filter ranges in the assigned CAN port of the XV-CAN200FD device. Valid used size of the arrays are depended by the “***i_wEXTFIDListSize***” parameter. The contents of each element are defined as below:
Element0: CAN ID from i_wEXTFID1[0] to i_wEXTFID2[0]
Element1: CAN ID from i_wEXTFID1[1] to i_wEXTFID2[1]
…
Element63: CAN ID from i_wEXTFID1[63] to i_wEXTFID2[63]

Return Value:

Return 0 means success, others means failure.

4.5.10 CANFD_GetCANEXTIDFilter

This function is used to get CAN filter function of “extended CAN/CAN FD ID ranges” parameter in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_GetCANEXTIDFilter(WORD i_wDevice_id, BYTE i_byCANPort, WORD* o_wEXTFID1, WORD* o_wEXTFID2);
C#
Int32 CANFD_GetCANEXTIDFilter(Uint16 i_wDevice_id, Byte i_byCANPort, [In, Out] Uint16[] o_wEXTFID1, [In, Out] Uint16[] o_wEXTFID2);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_wEXTFID1, *o_wEXTFID2***

[in/out] This point to arrays of the elements of extended CAN ID filter ranges in the assigned CAN port of the XV-CAN200FD device. Each array must reserve space for saving maximum 64 elements of CAN/CAN FD extended ID. Valid used size of the arrays are depended by the “**o_wEXTFIDListSize**” parameter in the **CANFD_GetCANGlobalFilter()** API function. The contents of each element are defined as below:
Element0: CAN ID from i_wEXTFID1[0] to i_wEXTFID2[0]
Element1: CAN ID from i_wEXTFID1[1] to i_wEXTFID2[1]
…
Element63: CAN ID from i_wEXTFID1[63] to i_wEXTFID2[63]

Return Value:

Return 0 means success, others means failure.

4.5.11 CANFD_GetCANStatus

This function is used to get CAN status, CAN Bus transmitted/received error counter and software buffer status in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++

```
int CANFD_GetCANStatus(WORD i_wDevice_id, BYTE i_byCANPort, DWORD *o_dwCANStatus, DWORD *o_dwErrCnt, DWORD *o_dwBufStatus);
```

C#

```
Int32 CANFD_GetCANStatus(Uint16 i_wDevice_id, Byte i_byCANPort, out Uint32 o_dwCANStatus, out Uint32 o_dwErrCnt, out Uint32 o_dwBufStatus);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_dwCANStatus***

[out] The CAN Bus status in the assigned CAN port of the XV-CAN200FD device. Please refer to appendix 5.3 for “CAN Status” definition.

****o_dwErrCnt***

[out] The CAN Bus transmitted/received error counter in the assigned CAN port of the XV-CAN200FD device. Please refer to appendix 5.4 for “CAN Error Counter” definition.

****o_dwBufStatus***

[out] The CAN Bus transmitted/received buffer status in the assigned CAN port of the XV-CAN200FD device.

Bit	Symbol	Value	Description
0	RX		CAN1/CAN2 receive software buffer status
		0	Receive software buffer under-run
		1	Receive software buffer overrun
1	TX		CAN1/CAN2 transmit software buffer status
		0	Transmit software buffer under-run

		1	Transmit software buffer overrun
3:2	-		Reserved
4	EW		CAN1/2 Error Warning status.
		0	Both error counters are below the Error Warning limit of 96
		1	At least one of error counter has reached the Error Warning limit of 96
5	EP		CAN1/2 Error passive status
		0	The CAN is in Error Active state.
		1	The CAN is in the Error Passive state
6	BO		CAN1/2 Bus Off status
		0	The CAN is not in Bus OFF state.
		1	The CAN is in the Bus OFF state
31:7	-	-	Reserved

Return Value:

Return 0 means success, others means failure.

4.5.12 CANFD_SetCANReInitialize

This function restarts the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_SetCANReInitialize(WORD i_wDevice_id, BYTE i_byCANPort);
C#
Int32 CANFD_SetCANReInitialize(Uint16 i_wDevice_id, Byte i_byCANPort;

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

Return Value:

Return 0 means success, others means failure.

4.6. Communication Functions

These functions are used to send/receive CAN/CAN FD messages through XV-CAN200FD

4.6.1 CANFD_SetCANTxMsg

This function is used to send a CAN/CAN FD message to the software transmitted buffer of the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_SetCANTxMsg(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode, DWORD i_dwID, BYTE i_byRTR, BYTE i_byFDF, BYTE i_byDlen, BYTE *i_byData);
C#
Int32 CANFD_SetCANTxMsg(Uint16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode, Uint32 i_dwID, Byte i_byRTR, Byte i_byFDF, Byte i_byDlen, [In, Out] Byte[] i_byData);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

I_byMode

[in] CAN message mode.
0: 2.0A, 11-bit CAN ID
1: 2.0B, 29-bit CAN ID

i_dwID

[in] CAN message ID parameter.
Valid Range:
2.0A (11-bit CAN ID) mode ➔ 0x000 ~ 0x7FF
2.0B (29-bit CAN ID) mode ➔ 0x00000000 ~ 0x1FFFFFFF

i_byRTR

[in] CAN message RTR (Remote Transmission Request) parameter.
0: use data frame
1: use remote frame. (No effect for CAN FD frame)

i_byFDF

[in] *CAN*/CAN FD message.
0: use normal CAN frame
1: use CAN FD frame

i_byDlen

[in] CAN/CAN FD frame data length parameter.
Normal CAN frame: Valid range: 0x0 ~ 0x8
CAN FD frame: Valid range: 0x0 ~ 0xF

i_byDlen (Hexadecimal)	*Frame data length (Decimal)*	*i_byDlen (Hexadecimal)*	*Frame data length (Decimal)*
0x0	*0*	*0x8*	*8*
0x1	*1*	*0x9*	*12*
0x2	*2*	*0xA*	*16*
0x3	*3*	*0xB*	*20*
0x4	*4*	*0xC*	*24*
0x5	*5*	*0xD*	*32*
0x6	*6*	*0xE*	*48*
0x7	*7*	*0xF*	*64*

****i_byData***

[in/out] This point to an user defined 8 (Normal CAN frame) / 64 (CAN FD frame) bytes array buffer for saving CAN/CAN FD message data parameter.

Return Value:

Return 0 means success, others means failure.

4.6.2 CANFD_GetCANRxMsg

This function is used to get a CAN/CAN FD message from the software received buffer of the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++

```
int CANFD_GetCANRxMsg(WORD i_wDevice_id, BYTE i_byCANPort, BYTE* o_byType, BYTE* o_byMode, DWORD* o_dwID, BYTE* o_byRTR, BYTE* o_byFDF, BYTE* o_byDlen, BYTE *o_byData, DWORD *o_dw_TimeStamp_s, DWORD *o_dw_TimeStamp_us);
```

C#

```
Int32 CANFD_GetCANRxMsg(Uint16 i_wDevice_id, Byte i_byCANPort, out Byte o_byType, out Byte o_byMode, out Uint32 o_dwID, out Byte o_byRTR, out Byte o_byFDF, out Byte o_byDlen, [In, Out] Byte[] o_byData, out Uint32 o_dw_TimeStamp_s, out Uint32 o_dw_TimeStamp_us);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_byType***

[out] Received messages format.
0: receive a stand/extended CAN/CAN FD message
1: receive a event message.
Event message format:
Mode: 1 (CAN 2.0B, 29-bit CAN ID)
ID: 0xEEEEEEE
RTR: 0 (No RTR)
FDF: 0
Dlen: 0x08
Data: D0~D3 ➔ CAN Bus status in little-endian format
(Please refer to appendix 7.3 for “CAN Status” definition.)

D4~D7 ➔ CAN Bus error counter in little-endian format (Please refer to appendix 7.4 for “CAN Error Counter” definition.)

o_byMode

[out] CAN message mode.
0: 2.0A, 11-bit CAN ID
1: 2.0B, 29-bit CAN ID

o_dwID

[out] CAN message ID parameter.
2.0A (11-bit CAN ID) CAN message ➔ 0x000 ~ 0x7FF
2.0B (29-bit CAN ID) CAN message ➔ 0x00000000 ~ 0x1FFFFFFF
Event message ➔ 0xEEEEEEE

o_byRTR

[out] CAN message RTR (Remote Transmission Request) parameter.
0: data frame
1: remote frame. (always 0 for CAN FD frame)

o_byFDF

[in] *CAN*/CAN FD message.
0: normal CAN frame
1: CAN FD frame

o_byDlen

[in] CAN/CAN FD frame data length parameter.
Normal CAN frame: Valid range: 0x0 ~ 0x8
CAN FD frame: Valid range: 0x0 ~ 0xF

o_byDlen (Hexadecimal)	*Frame data length (Decimal)*	*o_byDlen (Hexadecimal)*	*Frame data length (Decimal)*
0x0	*0*	*0x8*	*8*
0x1	*1*	*0x9*	*12*
0x2	*2*	*0xA*	*16*
0x3	*3*	*0xB*	*20*
0x4	*4*	*0xC*	*24*
0x5	*5*	*0xD*	*32*
0x6	*6*	*0xE*	*48*
0x7	*7*	*0xF*	*64*

***o_byData**

[in/out] This point to an user defined 64 bytes array buffer for saving CAN/CAN FD message data parameter.

***o_dw_TimeStamp_s**

[out] The timestamp of the received/event message.
Unit: second.

***o_dw_TimeStamp_us**

[out] The timestamp of the received/event message.
Unit: micro second.

Return Value:

Return 0 means success, others means failure.

4.6.3 CANFD_SetCANHWSendMode

This function is used to enable or disable CAN/CAN FD message sending in the assigned CAN port by using module hardware timer and it will be more precise than PC software timer.

Syntax:

C++
int CANFD_SetCANHWSendMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE i_byMode);
C#
Int32 CANFD_SetCANHWSendMode(Uint16 i_wDevice_id, Byte i_byCANPort, Byte i_byMode);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

I_byMode

[in] Enable or disable CAN/CAN FD message sending in the assigned CAN port by using module hardware timer.
0: disable CAN/CAN FD message sending by using module hardware timer.
1: enable CAN/CAN FD message sending by using module hardware timer.

Return Value:

Return 0 means success, others means failure.

4.6.4 CANFD_GetCANHWSendMode

This function is used to get the operating mode of CAN/CAN FD message sending in the assigned CAN port by using module hardware timer.

Syntax:

C++
int CANFD_GetCANHWSendMode(WORD i_wDevice_id, BYTE i_byCANPort, BYTE *o_byMode);
C#
Int32 CANFD_GetCANHWSendMode(Uint16 i_wDevice_id, Byte i_byCANPort, out Byte o_byMode);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_byMode***

[out] Operating mode of CAN/CAN FD message sending in the assigned CAN port by using module hardware timer.
0: disable.
1: enable.

Return Value:

Return 0 means success, others means failure.

4.6.5 CANFD_SetCANHWSendMsg

This function is used to set the CAN/CAN FD message sending in the assigned CAN port by using module hardware timer.

Syntax:

C++

```
int CANFD_SetCANHWSendMsg(WORD i_wDevice_id, BYTE i_byCANPort, BYTE
i_byMode, DWORD i_dwID, BYTE i_byRTR, i_byFDF, BYTE i_byDlen, BYTE
*i_byData, DWORD i_dwTimer, DWORD i_dwCounter);
```

C#

```
Int32 CANFD_SetCANHWSendMsg(Uint16 i_wDevice_id, Byte i_byCANPort, Byte
i_byMode, Uint32 i_dwID, Byte i_byRTR, Byte i_byFDF, Byte i_byDlen, [In, Out] Byte[]
i_byData, Uint32 i_dwTimer, Uint32 i_dwCounter);
```

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

I_byMode

[in] CAN message mode.
0: 2.0A, 11-bit CAN ID
1: 2.0B, 29-bit CAN ID

i_dwID

[in] CAN message ID parameter.
Valid Range:
2.0A (11-bit CAN ID) mode ➔ 0x000 ~ 0x7FF
2.0B (29-bit CAN ID) mode ➔ 0x00000000 ~ 0x1FFFFFFF

i_byRTR

[in] CAN message RTR (Remote Transmission Request) parameter.
0: use data frame
1: use remote frame. (No effect for CAN FD frame)

i_byDlen

[in] CAN/CAN FD frame data length parameter.
Normal CAN frame: Valid range: 0x0 ~ 0x8
CAN FD frame: Valid range: 0x0 ~ 0xF

i_byDlen (Hexadecimal)	*Frame data length (Decimal)*	*i_byDlen (Hexadecimal)*	*Frame data length (Decimal)*
0x0	*0*	*0x8*	*8*
0x1	*1*	*0x9*	*12*
0x2	*2*	*0xA*	*16*
0x3	*3*	*0xB*	*20*
0x4	*4*	*0xC*	*24*
0x5	*5*	*0xD*	*32*
0x6	*6*	*0xE*	*48*
0x7	*7*	*0xF*	*64*

****i_byData***

[in/out] This point to an user defined 8 (Normal CAN frame) / 64 (CAN FD frame) bytes array buffer for saving CAN/CAN FD message data parameter.

I_dwTimer

[in] Time period of the module hardware timer to send this CAN message.
Unit: 100 micro second

i_dwCounter

[in] Number of transmissions of the module hardware timer to send this CAN message.

Return Value:

Return 0 means success, others means failure.

4.6.6 CANFD_GetCANRxFramePerSec

This function is used to get the CAN Bus data flow in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_GetCANRxFramePerSec(WORD i_wDevice_id, BYTE i_byCANPort, WORD *o_wRxFPS);
C#
Int32 CANFD_GetCANRxFramePerSec(Uint16 i_wDevice_id, Byte i_byCANPort, out Uint16 o_wRxFPS);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_wRxFPS***

[out] The CAN Bus data flow in the assigned CAN port of the XV-CAN200FD device.

Return Value:

Return 0 means success, others means failure.

4.7. Software Buffer Functions

All the transmitted/received CAN messages will be saved in software buffer provided by CAN_FD library first. These related software functions are used to operate the software buffer of CAN_FD library.

4.7.1 CANFD_GetCANRxMsgCount

This function is used to get the count of received CAN/CAN FD messages in the software received buffer in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_GetCANRxMsgCount(WORD i_wDevice_id, BYTE i_byCANPort, DWORD *o_dwCount);
C#
Int32 CANFD_GetCANRxMsgCount(Uint16 i_wDevice_id, Byte i_byCANPort, out Uint32 o_dwCount);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

****o_dwCount***

[out] The count of received CAN/CAN FD messages in the software received buffer in the assigned CAN port of the XV-CAN200FD device.

Return Value:

Return 0 means success, others means failure.

4.7.2 CANFD_ClearCANRxBuf

This function is used to clear all the CAN/CAN FD messages in the software received buffer in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_ClearCANRxBuf(WORD i_wDevice_id, BYTE i_byCANPort);
C#
Int32 CANFD_ClearCANRxBuf(Uint16 i_wDevice_id, Byte i_byCANPort);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

Return Value:

Return 0 means success, others means failure.

4.7.3 CANFD_ClearCANTxBuf

This function is used to clear all the CAN/CAN FD messages in the software transmitted buffer in the assigned CAN port of the XV-CAN200FD device.

Syntax:

C++
int CANFD_ClearCANTxBuf(WORD i_wDevice_id, BYTE i_byCANPort);
C#
Int32 CANFD_ClearCANTxBuf(Uint16 i_wDevice_id, Byte i_byCANPort);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_byCANPort

[in] The assigned CAN port of the XV-CAN200FD device.
1: CAN1, 2: CAN2.

Return Value:

Return 0 means success, others means failure.

4.8. Other Functions

These functions are used to get the CAN_FD library information or helpful for users program.

4.8.1 CANFD_GetDllVersion

This function is used to get the version of CAN_FD library

Syntax:

C++
DWORD CANFD_GetDllVersion(void);
C#
Uint32 CANFD_GetDllVersion();

Parameter:

None.

Return Value:

Return the CAN_FD library version.
Value 1000000 (in decimal) ➔ CAN_FD library version: v1.0.0.0
Value 1000113 (in decimal) ➔ CAN_FD library version: v1.0.1.13

4.8.2 CANFD_GetFwVer

This function is used to get the firmware version of the XV-CAN200FD device

Syntax:

C++
Int CANFD_GetFwVer(WORD i_wDevice_id, WORD* o_wFwVer);
C#
Int32 CANFD_GetFwVer(Uint16 i_wDevice_id, out Uint16 o_wFwVer);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

****o_wFwVer***

[out] The firmware version of the XV-CAN200FD device.
Value 100 (in decimal) ➔ firmware version: v1.00

Return Value:

Return 0 means success, others means failure.

4.8.3 CANFD_SetSN

This function is used to set the BID (board ID) of the XV-CAN200FD device

Syntax:

C++
Int CANFD_SetSN(WORD i_wDevice_id, DWORD i_dwSN)
C#
Int32 CANFD_SetSN(Uint16 i_wDevice_id, Uint32 i_dwSN);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

I_dwSN

[in] The BID (board ID) of the XV-CAN200FD device.

Return Value:

Return 0 means success, others means failure.

4.8.4 CANFD_ResetModule

This function is used to reset the XV-CAN200FD device

Syntax:

C++
Int CANFD_ResetModule(WORD i_wDevice_id);
C#
Int32 CANFD_ResetModule(Uint16 i_wDevice_id);

Parameter:

i_wDevice_id

[in] The assigned device ID of the XV-CAN200FD device.

Return Value:

Return 0 means success, others means failure.

4.9. Return Codes

The return value is used to show the result of executing CAN_FD library functions. The following is the all return codes.

Error Code (hexadecimal)	Description
0x0	No error
0x1	OP field of the configuration command error
0x2	FC field of the configuration command error
0x3	DL field of the configuration command error
0x4	Fail to write data into device
0x10001	Invalid device
0x10002	Device already in used
0x10003	Device not exist
0x10004	Get device information error
0x10005	Invalid USB package size
0x10006	Write file fail
0x10007	USB Tx buffer overflow
0x1000A	Exceed maximum supported USB device
0x1000B	USB device not open
0x10100	Communication timeout
0x10101	Invalid CAN port number
0x10102	No data in CAN received buffer
0x10103	CAN transmitted buffer overflow
0x10104	CAN bit rate parameters not support
0x10105	CAN Filter ID list size parameter not support

4. Appendix

4.1. Revision History

This chapter provides revision history information to this document.
The table below shows the revision history.

Revision	Date	Description
1.0.0	2025/05	Initial issue
1.0.1	2026/02	Support LP-2841M

4.2. CAN Status Register

Bit	Symbol	Value	Description
2:0	LEC		**Last error code** These bits indicate the type of the last error to occur on the CAN bus. This bit field will be cleared when a message has been transferred without error. The bits in this bit field will be set upon a read access.
		0x0	**No error.**
		0x1	**Stuff error**: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
		0x2	**Form error**: A fixed format part of a received frame has the wrong format.
		0x3	**Ack Error**: The message transmitted by the M_CAN was not acknowledged by another node.
		0x4	**Bit1 Error**: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
		0x5	**Bit0 Error**: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
		0x6	**CRC Error**: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
		0x7	**Unused**: No CAN Bus event was detected
4:3	ACT		Activity. This register monitors the CAN communication state.
		0x0	**Synchronizing** – node is synchronizing on CAN communication.
		0x1	**Idle** – node is neither receiver nor transmitter.
		0x2	**Receiver** – node is operating as receiver
		0x3	**Transmitter** – node is operating as transmitter.
5	EP		Error passive
		0	The CAN controller is in the error active state.
		1	The CAN controller is in the error passive state as defined in the CAN 2.0 specification.
6	EW		Warning status
		0	Both error counters are below the Error Warning limit of 96
		1	At least one of error counter has reached the Error Warning limit of 96
7	BOFF		Bus off status
		0	The CAN module is not in bus off state.
		1	The CAN controller is in bus off state.
31:8	-	-	Reserved

4.3. CAN Error Counter Register

Bit	Symbol	Value	Description
7:0	TEC		Transmit error counter Current value of the transmit error counter (maximum value 255)
14:8	REC		Receive error counter Current value of the receive error counter (maximum value 127).
15	RP		Receive error passive
		0	Below error level. The receive counter is below the error passive level of 128
		1	At error level. The receive counter has reached the error passive level of 128
31:16	-	-	Reserved

4.4. Valid Data Phase Bit Rate

	Supported Data Phase Bit Rate (kbps)				
Items	0	1	2	3	4
0	10000.000	8571.429	7500.000	6666.667	6000.000
5	5454.545	5000.000	4615.385	4285.714	4000.000
10	3750.000	3529.412	3333.333	3157.895	3000.000
15	2857.143	2727.273	2608.696	2500.000	2400.000
20	2307.692	2222.222	2142.857	2068.966	2000.000
25	1935.484	1875.000	1818.182	1764.706	1714.286
30	1666.667	1621.622	1578.947	1538.462	1500.000
35	1463.415	1428.571	1395.349	1363.636	1333.333
40	1304.348	1276.596	1250.000	1224.49	1200.000
45	1176.471	1153.846	1132.075	1111.111	1090.909
50	1071.429	1052.632	1034.483	1016.949	1000.000
55	983.6066	967.7419	952.381	937.500	923.0769
60	909.0909	895.5224	882.3529	869.5652	857.1429
65	845.0704	833.3333	821.9178	810.8108	800.000
70	789.4737	779.2208	769.2308	759.4937	750.000
75	740.7407	731.7073	722.8916	714.2857	705.8824
80	697.6744	689.6552	681.8182	674.1573	666.6667
85	659.3407	652.1739	645.1613	638.2979	631.5789
90	625.000	618.5567	612.2449	606.0606	600.000
95	594.0594	588.2353	582.5243	576.9231	571.4286
100	566.0377	560.7477	555.5556	550.4587	545.4545
105	540.5405	535.7143	530.9735	526.3158	521.7391
110	517.2414	512.8205	508.4746	504.2017	500.000
115	495.8678	491.8033	487.8049	483.871	480.000
120	476.1905	472.4409	468.750	465.1163	461.5385
125	458.0153	454.5455	451.1278	447.7612	444.4444
130	441.1765	437.9562	434.7826	431.6547	428.5714
135	425.5319	422.5352	419.5804	416.6667	413.7931
140	410.9589	408.1633	405.4054	402.6846	400.000
145	397.351	394.7368	392.1569	389.6104	387.0968
150	384.6154	382.1656	379.7468	377.3585	375.000
155	372.6708	370.3704	368.0982	365.8537	363.6364
160	361.4458	359.2814	357.1429	355.0296	352.9412
165	350.8772	348.8372	346.8208	344.8276	342.8571

170	340.9091	338.9831	337.0787	335.1955	333.3333
175	331.4917	329.6703	327.8689	326.087	324.3243
180	322.5806	320.8556	319.1489	317.4603	315.7895
185	314.1361	312.500	310.8808	309.2784	307.6923
190	306.1224	304.5685	303.0303	301.5075	300.000
195	298.5075	297.0297	295.5665	294.1176	292.6829
200	291.2621	289.8551	288.4615	287.0813	285.7143
205	284.3602	283.0189	281.6901	280.3738	279.0698
210	277.7778	276.4977	275.2294	273.9726	272.7273
215	271.4932	270.2703	269.0583	267.8571	266.6667
220	265.4867	264.3172	263.1579	262.0087	260.8696
225	259.7403	258.6207	257.5107	256.4103	255.3191
230	254.2373	253.1646	252.1008	251.046	250.000
235	248.9627	247.9339	246.9136	245.9016	244.898
240	243.9024	242.915	241.9355	240.9639	240.000
245 ~ 290	…				
290	202.7027	202.0202	201.3423	200.6689	200.000
295 ~ 365	…				
365	161.7251	161.2903	160.8579	160.4278	160.000
370~390	…				
390	151.5152	151.1335	150.7538	150.3759	150.000
395~470	…				
470	126.0504	125.7862	125.523	125.261	125.000
475 ~ 490	…				
490	120.9677	120.7243	120.4819	120.2405	120.000
495 ~ 590	…				
590	100.6711	100.5025	100.3344	100.1669	100.000