InnoEX

Ethernet-CANbus

Windows Software Programming Guide

Ethernet-CANbus DLL Control APIs

1. Function List

This section provides the specifications of all Ethernet-CANbus functions and structures. All APIs use the naming convention **FintekCanbus_xxx** specific to below table:

ID	Function Name
1.1	FintekCanbus_Open
1.2	FintekCanbus_Close
1.3	FintekCanbus_SetBaudRate
1.4	FintekCanbus_GetBaudRate
1.5	FintekCanbus_SetId
1.6	FintekCanbus_SetFilter
1.7	FintekCanbus_ClearFilter
1.8	FintekCanbus_SetAcrAmrFilter
1.9	FintekCanbus_SetErrorFilter
1.10	FintekCanbus_GetErrorFilter
1.11	FintekCanbus_SetIer
1.12	FintekCanbus_GetIer
1.13	FintekCanbus_GetErrorCode
1.14	FintekCanbus_Send
1.15	FintekCanbus_Receive
1.16	FintekCanbus_ReceiveEx
1.17	Reserve
1.18	Reserve
1.19	Reserve
1.20	FintekCanbus_SetMode
1.21	FintekCanbus_GetMode
1.22	FintekCanbus_Reset
1.23	FintekCanbus_Create
1.24	FintekCanbus_Delete
1.25	FintekCanbus_Start
1.26	FintekCanbus_Stop

1.1 FintekCanbus_Open

```
long FintekCanbus_Open(IN char* sComPortNumber);
```

Function:

Open Canbus virtual COM Port.

Parameters:

sComPortNumber: COM port number.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the ErrorCodes of the document.

1.2 FintekCanbus_Close

```
long FintekCanbus_Close(IN char* sComPortNumber);
```

Function:

Close Canbus virtual COM Port.

Parameters:

sComPortNumber: COM port number.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.3 FintekCanbus_SetBaudRate

```
long FintekCanbus_SetBaudRate(IN char* sComPortNumber, IN DWORD BaudRate);
```

Function:

Set Canbus port Baud Rate.

Parameters:

sComPortNumber: COM port number.

BaudRate:

1M:	1000000
800K:	800000
500K:	500000
250K:	250000
100K:	100000
50K:	50000
20K:	20000
10K:	10000

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

How to use the Parameters "BaudRate" to input sample point (BTR value) settings for the CANBus controller:

You can use the parameter "BaudRate" to set a special BAUDRATE or specify a sample point for a specific application. For the setting method of sample point and BTR. Please refer to APPLICATION NOTE "Determination of Bit Timing Parameters for the CAN Controller SJA 1000 AN97046".

Parameters:

BaudRate:

bit[16]: Activate custom BAUDRATE sample point setting

bit[15:0]: Bit Timing Register (BTR) value

The following table shows the default value of BTR for each BAUDRATE of the controller:

CAN baudrate	Default BTR value (hex)	sample point%
1000 C027 75	800 C039 73.33	500 C127 75
250 C23S 75	125 C53A 75	100 C54D 75
50 CB4D 75	20 DD4D 75	10 FB4D 75

Example for the CANBus with clock frequency 24 MHz for the CAN controller, BTR=0xC009:

```
FintekCanbus_SetBaudRate("COM10", 0x1C009);
```

So, for this example, a BTR value of 0xC009 gives a CAN baudrate of 1000 Kbit / second.

1.4 FintekCanbus_GetBaudRate

```
long FintekCanbus_GetBaudRate(IN char* sComPortNumber, OUT DWORD* BaudRate);
```

Function:

Get Canbus port Baud Rate.

Parameters:

sComPortNumber: COM port number.

*BaudRate:

1M:1000000

800K:800000

500K:500000

250K:250000

100K:100000

50K:50000

20K:20000

10K:10000

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.5 FintekCanbus_SetId

```
long FintekCanbus_SetId (IN char* sComPortNumber, IN CanFrameFormat type, IN DWORD id);
```

Function:

Set Canbus id.

Parameters:

sComPortNumber: COM port number.

type: An CanFrameFormat enumerated type specifying the CAN protocol with 11bit or 29bit.

```
enum class CanFrameFormat {
    CP_29Bit,
    CP_11Bit
};
```

id: Specified the id.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.6 FintekCanbus_SetFilter

```
long FintekCanbus_SetFilter(IN char* sComPortNumber, IN DWORD pattern, IN DWORD mask);
```

Function:

Set Canbus port Filter. Default: pattern:0; mask:0, all frame will be received

Parameters:

sComPortNumber: COM port number.

pattern: Specified CAN ID pattern to filter.

mask: Specified the mask for filter.

Return Value:

If the function succeeds, the return value is filtering total count - 1 (0-14, MAX: 15). If the function fails, the return value is error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.7 FintekCanbus_ClearFilter

```
long FintekCanbus_ClearFilter(IN char* sComPortNumber);
```

Function:

Clear Canbus port Filter.

Parameters:

sComPortNumber: COM port number.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.8 FintekCanbus_SetAcrAmrFilter

```
long FintekCanbus_SetAcrAmrFilter(IN char* sComPortNumber, IN char cFilterMode, IN DWORD acr, IN DWORD amr);
```

Function:

Set Canbus port ACR/AMR HW Filter.

Parameters:

sComPortNumber: COM port number.

cFilterMode: 1: Single Filter, 0: Dual Filter.

acr: Specified CAN ID ACR (Acceptance Code Register).

amr: Specified the AMR (Acceptance Mask Register).

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

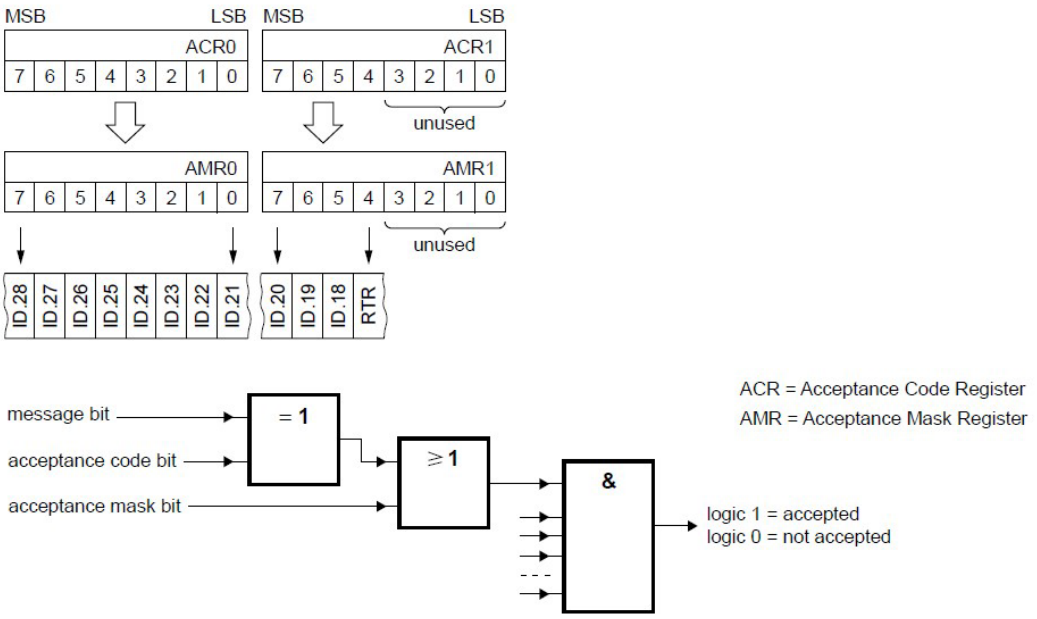
Notice:

With the help of the acceptance filter the CAN controller is able to allow passing of received messages to the RXFIFO only when the identifier bits of the received message are equal to the predefined ones within the acceptance filter registers. The acceptance filter is defined by the Acceptance Code Registers (ACR) and the Acceptance Mask Registers (AMR). The bit patterns of messages to be received are defined within the acceptance code registers.

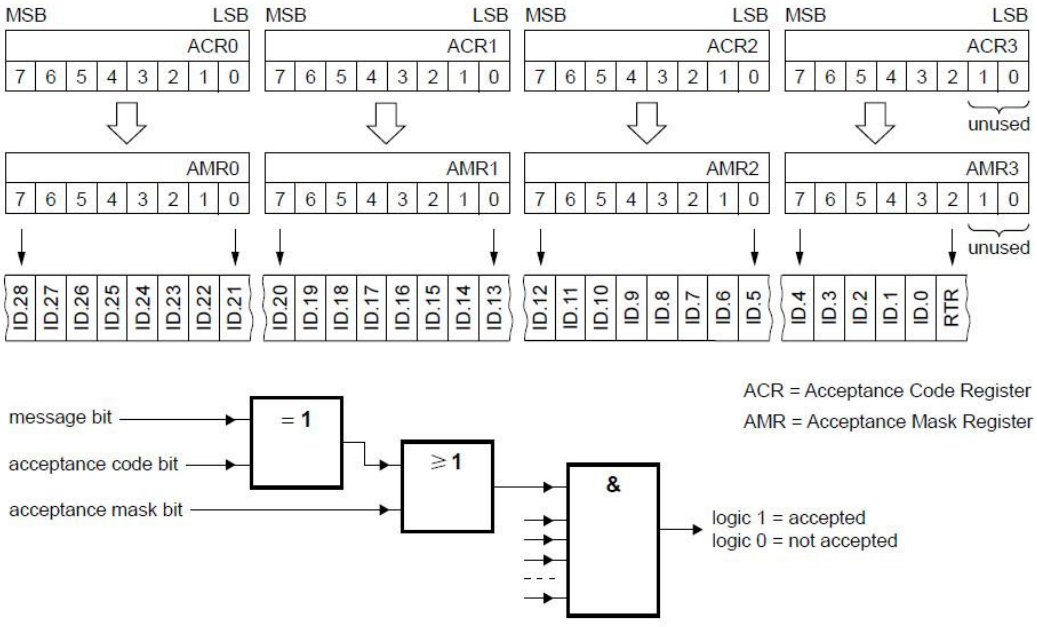
The corresponding acceptance mask registers allow to define certain bit positions to be “don’t care”.

Two different filter modes are selectable within the mode:

1. Single filter mode (cFilterMode is 1).

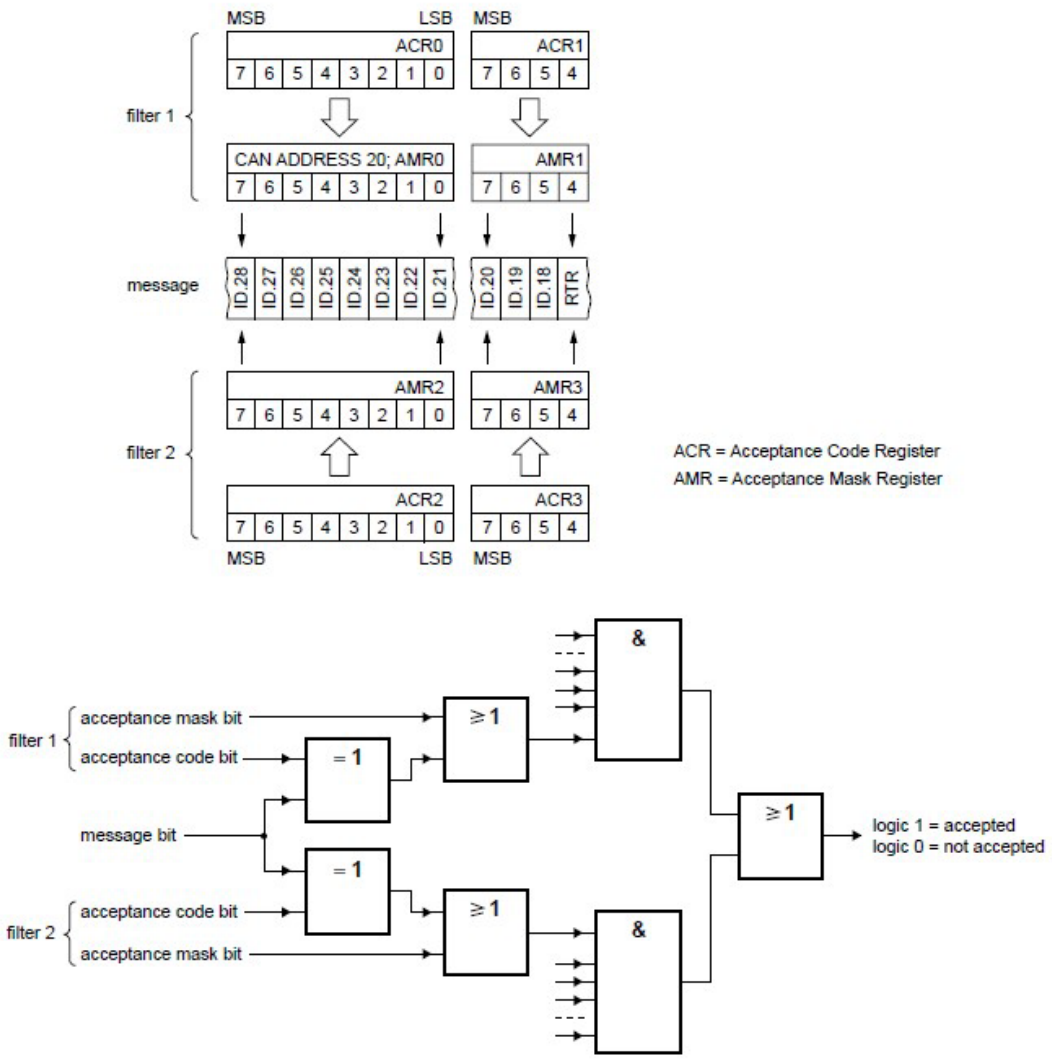


Single filter configuration, receiving standard frame messages.

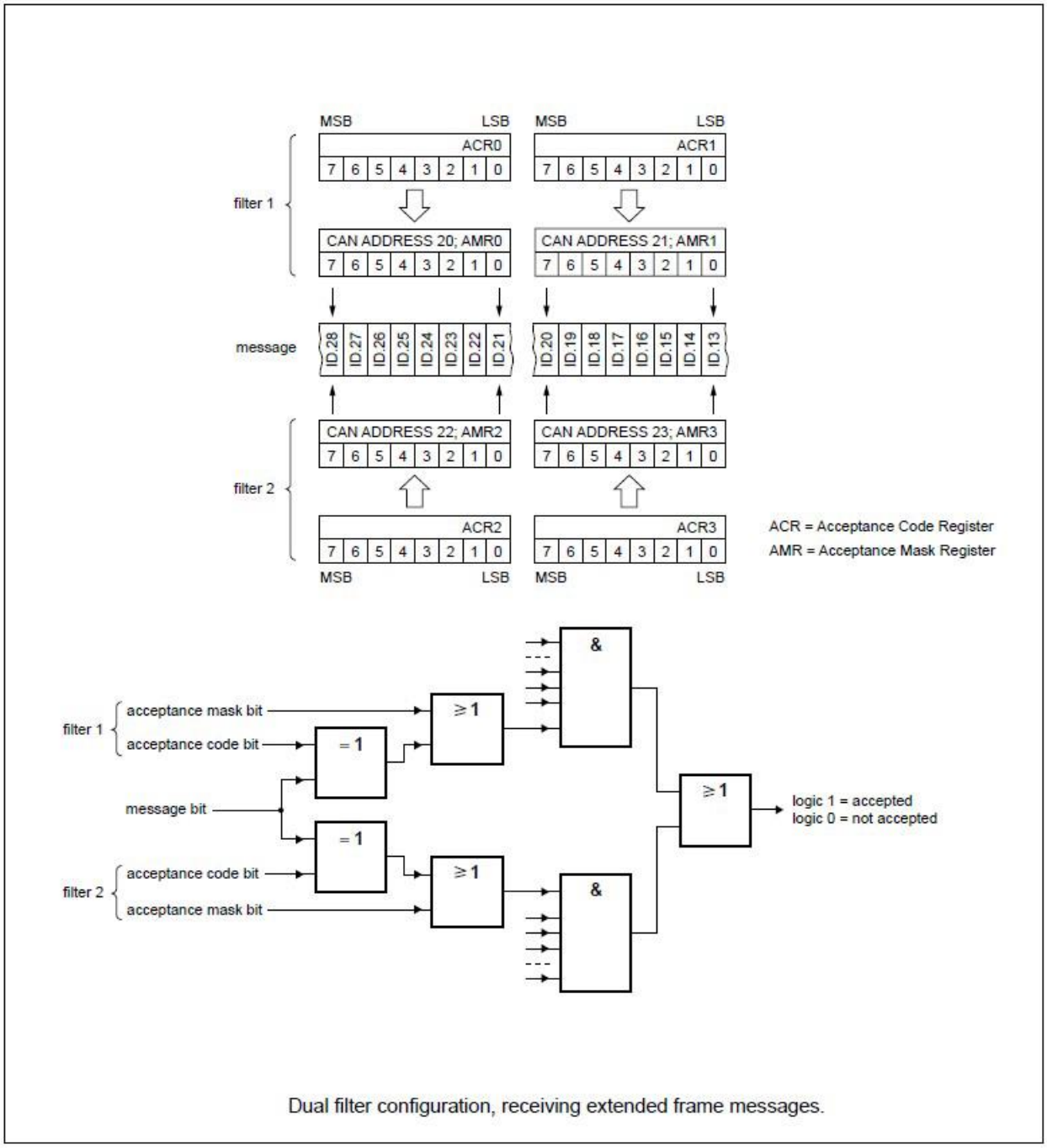


Single filter configuration, receiving extended frame messages.

2. Dual filter mode (cFilterMode is 0).



Dual filter configuration, receiving standard frame messages.



Dual filter configuration, receiving extended frame messages.

1.9 FintekCanbus_SetErrorFilter

```
long FintekCanbus_SetErrorFilter(IN char* sComPortNumber, IN UCHAR errorfilter);
```

Function:

Set Canbus port error or warning report filter. Parameter errorfilter allow to define certain bit positions to be ”don’t care”.

Parameters:

sComPortNumber: COM port number.

errorfilter: Set 0 to ignore the Error/Warning Bit report.

BIT7: Bus Error

BIT6: Arbitration Lost

BIT5: Error Passive

BIT4: Wake-Up

BIT3: Data Overrun

BIT2: Error Warning

BIT0-1: Reserved. Must to be 0

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.10 FintekCanbus_GetErrorFilter

```
long FintekCanbus_GetErrorFilter(IN char* sComPortNumber, OUT UCHAR* errorfilter);
```

Function:

Get Canbus port error or warning report filter value.

Parameters:

sComPortNumber: COM port number.

*errorfilter: Error/Warning Bit report value, 0 is ignore.

BIT7: Bus Error

BIT6: Arbitration Lost

BIT5: Error Passive

BIT4: Wake-Up

BIT3: Data Overrun

BIT2: Error Warning

BIT0-1: Reserved. Must to be 0

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.11 FintekCanbus_SetIer

```
long FintekCanbus_SetIer(IN char* sComPortNumber, IN UCHAR ier);
```

Function:

Set Canbus port interrupt enable register (IER). The function allows to enable/disable different types of interrupt sources

Parameters:

sComPortNumber: COM port number.

ier: Set 1 to enable the interrupt Bit report.

BIT7: Bus Error interrupt

BIT6: Arbitration Lost interrupt

BIT5: Error Passive interrupt

BIT4: Wake-Up interrupt

BIT3: Data Overrun interrupt

BIT2: Error Warning interrupt

BIT0-1: Reserved. Must to be 1

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.12 FintekCanbus_GetIer

```
long FintekCanbus_GetIer(IN char* sComPortNumber, OUT UCHAR* ier);
```

Function:

Get Canbus port interrupt enable register (IER).

Parameters:

sComPortNumber: COM port number.

*ier: The interrupt Bit report value, 0 is disable, 1 is enable.

BIT7: Bus Error interrupt

BIT6: Arbitration Lost interrupt

BIT5: Error Passive interrupt

BIT4: Wake-Up interrupt

BIT3: Data Overrun interrupt

BIT2: Error Warning interrupt

BIT0-1: Reserved. Must to be 1

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.13 FintekCanbus_GetErrorCode

```
long FintekCanbus_GetErrorCode(IN char* sComPortNumber, OUT DWORD* ErrorCode);
```

Function:

Get Canbus port error code value.

Parameters:

sComPortNumber: COM port number.

*ErrorCode:

bit [31:24]: Receive Error Counter

bit [23:16]: Transmit Error Counter

bit[15:8]: Error Code Capture

bit[7]: Bus Error

bit[6]: Arbitration Lost

bit[5]: Error Passive

bit[3]: Data Overrun

bit[2]: Error Warning

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.14 FintekCanbus_Send

```
long FintekCanbus_Send(IN char* sComPortNumber, CanFrameInfor* msg);
```

Function:

Send Canbus frame to virtual COM Port.

Parameters:

sComPortNumber: COM port number.

msg: The send frame, include frame format, frame id, frame length and frame data.

```
struct CanFrameInfor {
    CanFrameFormat type;
    BYTE rtr;
    DWORD id;
    BYTE data_len;
    BYTE data[CANBUS_MAX_DATA_SIZE];
};
enum class CanFrameFormat {
    CP_29Bit,
    CP_11Bit
};
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.15 FintekCanbus_Receive

```
long FintekCanbus_Receive(IN char* sComPortNumber, IN CanFrameInforProc callback);
```

Function:

Receive Canbus frame to virtual COM Port.

Parameters:

sComPortNumber: COM port number.

callback: `typedef void(CALLBACK *CanFrameInforProc) (long error, CanFrameInfor* msg);`

```
struct CanFrameInfor {
    CanFrameFormat type;
    BYTE rtr;
    DWORD id;
    BYTE data_len;
    BYTE data[CANBUS_MAX_DATA_SIZE];
};
enum class CanFrameFormat {
    CP_29Bit,
    CP_11Bit
};
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the callback parameter “error” is nonzero and the error code generated by it. You can find more information about error codes at the section ErrorCodes of the document.

1.16 FintekCanbus_ReceiveEx

```
long FintekCanbus_ReceiveEx(IN char* sComPortNumber, IN CanFrameInforProc callback);
```

Function:

Receive Canbus frame to virtual COM Port.

Parameters:

sComPortNumber: COM port number.

callback:

```
typedef void(CALLBACK *CanFrameInforProcEx)(long error, CanFrameInforEx* msg);
struct CanFrameInforEx {
    CanFrameFormat type;
    INT com_number;
    BYTE rtr;
    DWORD id;
```

```
    BYTE data_len;
    BYTE data[CANBUS_MAX_DATA_SIZE];
};
enum class CanFrameFormat {
    CP_29Bit,
    CP_11Bit
};
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the callback parameter “error” is nonzero and the error code generated by it. You can find more information about error codes at the section ErrorCodes of the document.

1.20 FintekCanbus_SetMode

```
long FintekCanbus_SetMode(IN char* sComPortNumber, IN UCHAR mode);
```

Function:

Set Canbus Mode. The function allows to change the behavior of the CANBus controller.

Parameters:

sComPortNumber: COM port number.

mode: Set 1 to enable the Canbus Mode.

BIT3: Acceptance Filter Mode: 1: Single Filter, 0: Dual Filter.

BIT1: Listen Only Mode: 1 is enable, 0 is disable.

BIT0, 2, 4-7: Reserved.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.21 FintekCanbus_GetMode

```
long FintekCanbus_GetMode(IN char* sComPortNumber, OUT UCHAR* mode);
```

Function:

Get Canbus Mode.

Parameters:

sComPortNumber: COM port number.

* mode: The Mode Bit report value.

BIT3: Acceptance Filter Mode: 1: Single Filter, 0: Dual Filter.

BIT1: Listen Only Mode: 1 is enable, 0 is disable.

BIT0, 2, 4-7: Reserved.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.22 FintekCanbus_Reset

```
long FintekCanbus_Reset(IN char* sComPortNumber, IN UCHAR mode);
```

Function:

Reset Canbus port.

Parameters:

sComPortNumber: COM port number.

mode: Reserved. Must to be 0

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.23 FintekCanbus_Create

```
long FintekCanbus_Create(IN char* sComPortNumber);
```

Function:

Initializes (create) canbus port but does not start it. Please use function FintekCanbus_Start to startup. The difference from FintekCanbus_Open function is that includes initialization and startup.

FintekCanbus_Open = FintekCanbus_Create + FintekCanbus_Start

Parameters:

sComPortNumber: COM port number.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.24 FintekCanbus_Delete

```
long FintekCanbus_Delete(IN char* sComPortNumber);
```

Function:

Uninitialized (Delete) canbus port. This call is a reciprocal to FintekCanbus_Create.

Parameters:

sComPortNumber: COM port number.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.25 FintekCanbus_Start

```
long FintekCanbus_Start(IN char* sComPortNumber);
```

Function:

Start canbus port. You must initialize canbus before using this function. Please refer to the function FintekCanbus_Create description.

Parameters:

sComPortNumber: COM port number.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

1.26 FintekCanbus_Stop

```
long FintekCanbus_Stop(IN char* sComPortNumber);
```

Function:

Stop canbus port. This call is a reciprocal to FintekCanbus_Start

Parameters:

sComPortNumber: COM port number.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. You can find more information about error codes at the section ErrorCodes of the document.

2. ErrorCodes

The below table lists errors that Ethernet-CANBus functions API returns in response to calls.

ErrorCode	Description
0x80000005	CAN number ERROR
0x80000007	Thread ERROR
0x8005FFFF	FATAL ERROR (bus off)
0x80050FFF	CAN BUFFER FULL
0x80060002	INVALID HANDLE VALUE
0x80060003	No response from device
0x80060008	CAN function called fail
0x80060101	CAN communication fail
0x80060201	Write fail
0x80060202	Read fail
Other	CAN ERROR: bit [31:24]: Receive Error Counter bit [23:16]: Transmit Error Counter bit [15:8]: Error Code Capture bit [7]: Bus Error bit [6]: Arbitration Lost bit [5]: Error Passive bit [3]: Data Overrun bit [2]: Error Warning___