Application Note

CP_AN_002

CP-CANFD Programming Guide

Version 1.0

Issue Date: 10-06-2024

This application note describes technical information and instructions to program CP-CANFD Adapter

Connective Peripherals Pte Ltd
1 Tai Seng Avenue, Tower A, #03-07 Singapore 536464
Tel.: +65 67430980
E-Mail (Support): support@connectiveperipherals.com
Web: www.connectiveperipherals.com/products

Table of Contents

1 General

1.1 DLL Installation

There is no specific DLL installer. The DLL, LIB and Header files are typically copied to an application project directory. Different programming languages and compiler configurations require different locations. Consult your programming environment documentation for the correct locations to place the files contained within the distribution.

Following is a simple example project generated with Visual Studio.

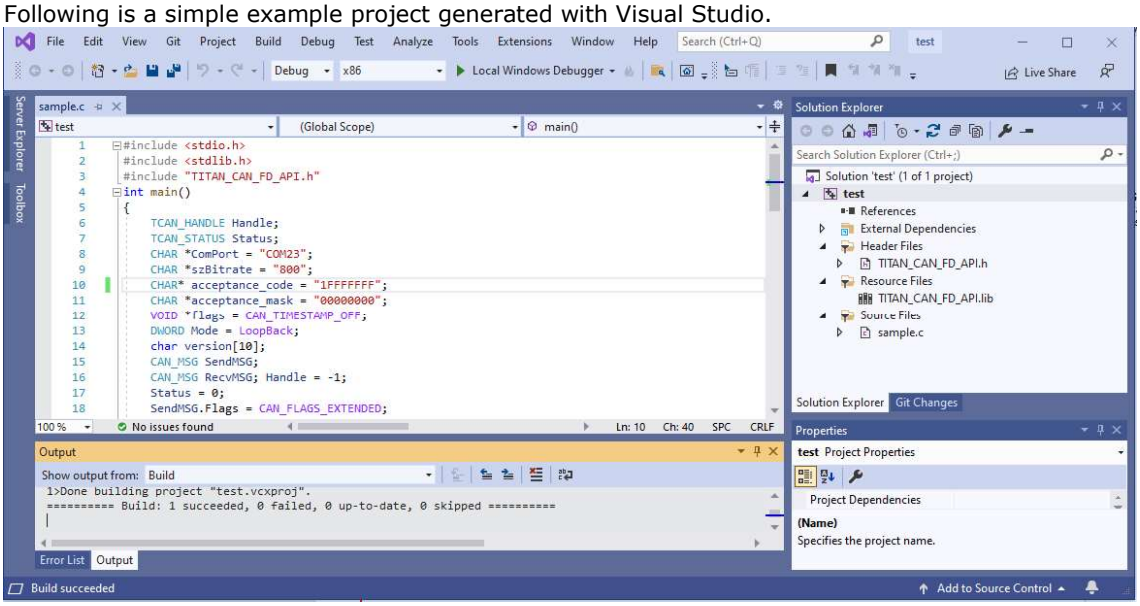

Figure 1: Sample Project

The DLL library must be installed.

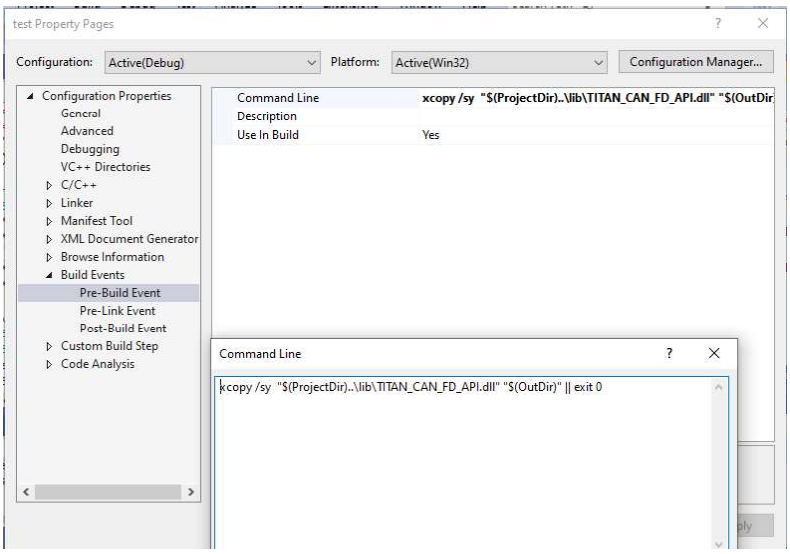

Figure 2: Project Configuration

1.2 Supported Products Information

This programming guide covers the following products only:

- CP-CANFD-1P USB to single CAN FD Adapter; Non-Isolation version
- CP-CANFD-1P-ISO USB to single CAN FD Adapter; Isolation version
- CP-CANFD-2P USB to dual CAN FD Adapter; Non-Isolation version
- CP-CANFD-2P-ISO USB to dual CAN FD Adapter; Isolation version

2 Application Programming Interface

2.1 Functions

2.1.1 CAN_OPEN

Summary

Open a channel to the CAN device.

Definition

TCAN_HANDLE CAN_Open(CHAR *ComPort, CHAR *szBitrate, CHAR *acceptance_code, CHAR *acceptance_mask, UCHAR flags , DWORD Mode);

Parameters

ComPort	CAN device Serial port name
szBitrate	10 = 10Kbps 20 = 20Kbps 50 = 50Kbps 100 = 100Kbps 125 = 125Kbps 250 = 250Kbps 500 = 500Kbps 800 = 800Kbps 1000 = 1000Kbps
acceptance_code	11-bit (0x000 to 0x7FF) or 29-bit (0x00000000 to 0x1FFFFFFF) code used in filtering specific or ranges of CAN messages. Default is to pass all frames (Acceptance Filter = 0x7FF for standard messages and 0x1FFFFFFF for extended messages).
acceptance_mask	11-bit (0x000 to 0x7FF) or 29-bit (0x00000000 to 0x1FFFFFFF) code used in filtering specific or ranges of CAN messages. A value of "0" in a bit location indicates that the bit location ID value is to be ignored when filtering messages. Default is to pass all frames (Acceptance Mask = 0x000 for standard messages and 0x00000000 for extended messages)
flags	CAN_TIMESTAMP_ON = The timestamp function will be enabled by the CAN device. CAN_TIMESTAMP_OFF = The timestamp function will be disabled.
Mode	CAN_MODE_NORMAL = the normal operation mode. CAN_MODE_LISTENONLY = the listen only mode.

CAN_MODE_LOOPBACK = the loop back mode, the CAN device will return messages you send.

Return value

CAN_HANDLE	Handle to the device if successful.
<= 0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
CHAR *ComPort = "COM11";
CHAR *szBitrate = "800";
CHAR *acceptance_code= "1FFFFFFF"; // Default is to pass all frames
CHAR * acceptance_mask= "00000000"; // Default is to pass all frames
UCHAR flags=CAN_TIMESTAMP_OFF;
DWORD Mode = Normal;
Handle=CAN_Open(ComPort,szBitrate,acceptance_code,acceptance_mask,flag,Mode);
```

Code Snippet 1 – Example for CAN_Open

2.1.2 CANFD_Open

Summary

Open a channel to the CAN FD device.

Definition

TCAN_HANDLE CANFD_Open(CHAR *ComPort, CHAR *szBitrate, CHAR *FDBitrate, CAN_Filters STDfilter, CAN_Filters EXDfilter, UCHAR flags, UCHAR Mode)

Parameters

ComPort	CAN device Serial port name
szBitrate	10 = 10Kbps 20 = 20Kbps 50 = 50Kbps 100 = 100Kbps 125 = 125Kbps 250 = 250Kbps 500 = 500Kbps 800 = 800Kbps 1000 = 1000Kbps
FDBitrate	100 = 100Kbps 125 = 125Kbps 250 = 250Kbps 500 = 500Kbps 800 = 800Kbps

	1000 = 1000Kbps 2000 = 2000Kbps 5000 = 5000Kbps 8000= 8000Kbps
STDfilter	Standard CAN_Filters structure
EXDfilter	Extended CAN_Filters structure
flags	CAN_TIMESTAMP_ON = The timestamp function will be enabled by the CAN device. CAN_TIMESTAMP_OFF = The timestamp function will be disabled.
Mode	CAN_MODE_NORMAL = the normal operation mode. CAN_MODE_LISTENONLY = the listen only mode. CAN_MODE_LOOPBACK = the loop back mode, the CAN device will return messages you send.

Return value

CAN_HANDLE	Handle to the device if successful.
<= 0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
CHAR *ComPort = "COM11";
CHAR *szBitrate = "800";
CHAR * FDBitrate = "5000";
CAN_Filters STDfilter=0; // Default is to pass all frames
CAN_Filters EXDfilter=0; // Default is to pass all frames
UCHAR flags=CAN_TIMESTAMP_OFF;
DWORD Mode = CAN_MODE_NORMAL;

Handle=CANFD_Open(ComPort,szBitrate,FDBitrate,FDBitrate,EXDfilter,flag,Mode);
CANFD_Start(Handle);
```

Code Snippet 2 – Example for CANFD_Open

2.1.3 CANFD_Start

Summary

Start the CAN FD channel with Handle

Definition

TCAN_STATUS CANFD_Start(TCAN_HANDLE Handle);

Parameters

Handle	the handle of the CAN channel which will be started

Return value

=1	CAN channel is started successfully.
<0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
Status = CANFD_Start(Handle);
```

Code Snippet 3 – Example for CANFD_Start

2.1.4 CAN_OpenOnly

Summary

Open the CAN channel without config.

Definition

TCAN_HANDLE CAN_OpenOnly(CHAR *ComPort);

Parameters

ComPort	CAN device Serial port name

Return value

TCAN _HANDLE	Handle to the device if successful.
<=0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
CHAR *ComPort = "COM11";
Handle = CAN_ OpenOnly (ComPort);
```

Code Snippet 4 – Example for CAN_OpenOnly

2.1.5 CAN_Close

Summary

Close the CAN channel with Handle

Definition

TCAN_STATUS CAN_Close(TCAN_HANDLE Handle);

Parameters

Handle	the handle of the CAN channel which will be closed

Return value

=1	CAN channel is closed successfully.
<0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
Status = CAN_Close(Handle);
```

Code Snippet 5 – Example for CAN_Close

2.1.6 CAN_Write

Summary

Write a message to the open channel with handle. Message is contained within the message structure.

Definition

TCAN_STATUS CAN_Write(TCAN_HANDLE Handle, CAN_MSG *Buf);

Parameters

Handle	handle returned by CAN_Open
Buf	transmitted message frame structure

Remarks

```
// Message flags
    #define CAN_FLAGS_STANDARD (1 << 0) // Stand CAN ID
    #define CAN_FLAGS_EXTENDED (1 << 1) // Extended CAN ID
    #define CAN_FLAGS_REMOTE (1 << 2) // Remote frame
// CAN Frame
    typedef struct
    {
        UINT32 Id;
        UINT8 Size;
        UINT8 Data[8];
        UINT8 Flags;
        UINT16 Timestamp;
    } CAN_MSG;
```

Code Snippet 6 – Message Frame Structure

Return value

=1	Writing transfer was successful.
<0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
CAN_MSG msg;
msg.Flags = CAN_FLAGS_STANDARD;

msg.Id = 0x111;
msg.Size = 5
msg.Data[0] = 0x11;
msg.Data[1] = 0x11;
msg.Data[2] = 0x11;
msg.Data[3] = 0x11;
msg.Data[4] = 0x11;
msg.Data[5] = 0x11;
msg.Data[6] = 0x11;
msg.Data[7] = 0x11;
Status=CAN_Write(Handle, &msg);
```

Code Snippet 7 – Example for CAN_Write

2.1.7 CAN_Read

Summary

Read a message from the open channel with handle. Message is contained within the message structure.

Definition

TCAN_STATUS CAN_Read(TCAN_HANDLE Handle, CAN_MSG *Buf);

Parameters

Handle	handle returned by CAN_Open
Buf	received message frame structure

Remarks

```
// Message flags
    #define CAN_FLAGS_STANDARD (1 << 0) // Stand CAN ID
    #define CAN_FLAGS_EXTENDED (1 << 1) // Extended CAN ID
    #define CAN_FLAGS_REMOTE (1 << 2) // Remote frame
// CAN Frame
    typedef struct
    {
        UINT32 Id;
        UINT8 Size;
        UINT8 Data[8];
        UINT8 Flags;
        UINT16 Timestamp;
    } CAN_MSG;
```

Code Snippet 8 – Message Frame Structure

Return value

=1	Reading transfer was successful.
<0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
CAN_MSG msg;
Status=CAN_Read( Handle, &msg );
```

Code Snippet 9 – Example for CAN_Read

2.1.8 CAN_Status

Summary

Retrieve adaptor status for channel with handle.

Definition

TCAN_STATUS CAN_Status(TCAN_HANDLE Handle);

Parameters

Handle	handle returned by CAN_Open

Return value

>0	Status value with bit values in 2.2.5 CAN_STATUS

Example

```
Status = CAN_Status ( Handle );
```

Code Snippet 10 – Example for CAN_Status

2.1.9 CAN_Version

Summary

Retrieve device firmware version with handle.

Definition

TCAN_STATUS CAN_Version(TCAN_HANDLE Handle, CHAR *Buf);

Parameters

Handle	handle returned by CAN_Open
Buf	device firmware version

Return value

=1	Version retrieved successfully.
<0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
CHAR Versioninfo[10];
Status=CAN_Version( Handle, Versioninfo );
```

Code Snippet 11 – Example for CAN_Version

2.1.10 CAN_Flush

Summary

Clears the buffers for device with handle.

Definition

TCAN_STATUS CAN_Flush(TCAN_HANDLE Handle);

Parameters

Handle	handle returned by CAN_Open

Return value

=1	Buffers cleared successfully.
<0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
Status = CAN_Flush( Handle );
```

Code Snippet 12 – Example for CAN_Flush

2.1.11 CAN_SN

Summary

Get the Serial number of the device with handle.

Definition

TCAN_STATUS CAN_SN(TCAN_HANDLE Handle, CHAR *buf);

Parameters

Handle	handle returned by CAN_Open
buf	device serial number

Return value

=1	Serial number retrieved successfully.
<0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
Status = CAN_SN( Handle, SerialNumber );
```

Code Snippet 13 – Example for CAN_SN

2.1.12 CAN_SetRcvEvent

Summary

Set the event of the device with handle.

Definition

TCAN_STATUS CAN_SetRcvEvent(TCAN_HANDLE Handle, HANDLE Event);

Parameters

Handle	handle returned by CAN_Open
Event	In Windows an event handle of the type HANDLE

Return value

=1	Event set successfully.

<0	Error as noted in 2.2.4 CAN_ERRORS

Example

```
HANDLE hEvent;
DWORD dwRetCode;
hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
CAN_SetRcvEvent(handle, hEvent);
dwRetCode = WaitForSingleObject(hEvent, INFINITE);
switch(dwRetCode)
{
    case WAIT_OBJECT_0 :
        // a CAN frame arrived
        break;
    default:
        // probe for error
}
CloseHandle(hEvent);
```

Code Snippet 14 – Example for CAN_SetRcvEvent

2.2 Types and Structures

2.2.1 CAN_HANDLE

```
typedef int TCAN_HANDLE;
```

Code Snippet 15 – CAN_HANDLE

2.2.2 CAN_MODE

```
CAN_MODE_NORMAL
CAN_MODE_LISTENONLY
CAN_MODE_LOOPBACK
```

Code Snippet 16 – CAN_MODE

2.2.3 CAN_TIMESTAMP

```
CAN_TIMESTAMP_OFF
CAN_TIMESTAMP_ON
```

Code Snippet 17 – CAN_TIMESTAMP

2.2.4 CAN_ERRORS

```
CAN_ERR_OK
CAN_ERR_ERR
CAN_ERR_OPEN_CHANNEL
CAN_ERR_PARAMETER
CAN_ERR_NOT_OPEN
CAN_ERR_READ_NO_MSG
```

Code Snippet 18 – CAN_ERRORS

2.2.5 CAN_STATUS

```
typedef int TCAN_STATUS;
CAN_STATUS_BOFF
CAN_STATUS_EWARN
CAN_STATUS_EPASS
CAN_STATUS_LEC_NO
CAN_STATUS_LEC_STUFF_ERR
CAN_STATUS_LEC_FORM_ERR
CAN_STATUS_LEC_ACK_ERR
CAN_STATUS_LEC_BIT1_ERR
CAN_STATUS_LEC_BIT0_ERR
```

Code Snippet 19 – CAN_STATUS

2.2.6 MSG Flags

```
CAN_FLAGS_STANDARD
CAN_FLAGS_EXTENDED
CAN_FLAGS_REMOTE
CAN_FLAGS_TIMESTAMP
CAN_FLAGS_FD
CAN_FLAGS_BRS
```

Code Snippet 20 – MSG Flags

2.2.7 CAN_MSG

```
typedef struct
{
    UINT32 Id;
    UINT8 Size;
    UINT8 Data[64];
    UINT8 Flags;
    UINT16 Timestamp;
} CAN_MSG;
```

Code Snippet 21 – CAN_MSG

2.2.8 FILTER

```
typedef struct {
    UINT32 Filter;
    UINT32 Mask;
} Filter;
```

Code Snippet 22 – FILTER

2.2.9 CAN_FILTERS

```
typedef struct {
    Filter Filters[20];
    INT NUM;
} CAN_Filters;
```

Code Snippet 23 – CAN_FILTERS

3 Example Code

```
#include <stdio.h>
#include <stdlib.h>
#include "CAN_API.h"
int main() {
    TCAN_HANDLE Handle;
  TCAN_STATUS Status;
  CHAR *ComPort = "COM23";
  CHAR *szBitrate = "800";
  CHAR *acceptance_code = "1FFFFFFF";
  CHAR *acceptance_mask = "00000000";
  VOID *flags = CAN_TIMESTAMP_OFF;
  DWORD Mode = LoopBack;

  char version[10];
  CAN_MSG SendMSG;
  CAN_MSG RecvMSG;
  Handle = -1;
  Status = 0;

  SendMSG.Flags = CAN_FLAGS_EXTENDED;
  SendMSG.Id = 0x12345678;
  SendMSG.Size = 8;
  SendMSG.Data[0] = 0x11;
  SendMSG.Data[1] = 0x22;
  SendMSG.Data[2] = 0x33;
  SendMSG.Data[3] = 0x44;
  SendMSG.Data[4] = 0x55;
  SendMSG.Data[5] = 0x66;
  SendMSG.Data[6] = 0x77;
  SendMSG.Data[7] = 0x88;

  Handle = CAN_Open ( ComPort, szBitrate, acceptance_code, acceptance_mask,
flags, Mode );

  printf ( "handle= %d\n", Handle );
  if ( Handle < 0 ) {
      return 0;
  }
    memset ( version, 0, sizeof ( char ) * 10 );
    Status = CAN_Flush ( Handle );
  Status = CAN_Version ( Handle, version );

    if ( Status == CAN_ERR_OK ) {
        printf ( "Version : %s\n", version );
  }

    Status = CAN_Write ( Handle, &SendMSG );
    if ( Status == CAN_ERR_OK ) {
        printf ( "Write Success\n" );
  }
```

```
        while ( 1 ) {
         Status = CAN_Read ( Handle, &RecvMSG );
         if ( Status == CAN_ERR_OK ) {
             printf ( "Read ID=0x%X, Type=%s, DLC=%d, FrameType=%s, Data=",
             RecvMSG.Id,( RecvMSG.Flags & CAN_FLAGS_STANDARD ) ? "STD" : "EXT",
             RecvMSG.Size,( RecvMSG.Flags & CAN_FLAGS_REMOTE ) ? "REMOTE" :
"DATA" );
             for ( int i = 0; i < RecvMSG.Size; i++ ) {
                  printf ( "%X,", RecvMSG.Data[i] );
             }
             break;
         }
    }
    Status = CAN_Close ( Handle );
    printf ( "Test finish\n" );
    return 0;
}
```

Code Snippet 24 – Example Code

4 Contact Information

Global Headquarters – Singapore

Connective Peripherals Pte Ltd
1 Tai Seng Avenue, Tower A,
#03-07
Singapore 536464

Tel: +65 67430980

E-Mail (Sales) sales@connectiveperipherals.com
E-Mail (Support) support@connectiveperipherals.com
Web Site URL http://www.connectiveperipherals.com
Web Shop URL http://www.connectiveperipherals.com

Appendix A – References

Document References

Acronyms and Abbreviations

Terms	Description
CAN	Controller Area Network
CANFD	Controller Area Network Flexible Data-rate
DLL	Dynamic Link Library

Appendix B – List of Tables/Figures/Code Snippets

List of Tables

NA

List of Figures

List of Code Snippet

Appendix C – Revision History

Document Title: CP_AN_002 CP-CANFD Programming Guide
Document Reference No.: CP_000099
Clearance No.: CP#080
Product Page: https://connectiveperipherals.com/
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	10-06-2024