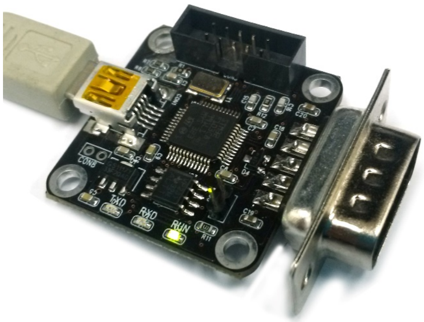
USB2CAN

USB2CAN-UART

USB2CAN-FIFO



API Reference Manual

http://ntrexgo.com

설명서에 포함된 정보는 정확하고 신뢰성이 있는 내용입니다. 그러나 출판 당시 발견되지 않은 오류가 있을 수 있으니 사용자는 자신의 제품 검증을 수행하시기 바라며, 전적으로 설명서에 포함된 정보에 의존하지 마시기 바랍니다.

1 소개

LinKerS USB2CAN은 Host PC에서 USB 포트를 이용하여 CAN bus에 쉽게 접근할 수 있게 해주는 USB to CAN BUS 어댑터 모듈입니다.

이 문서는 USB2CAN 컨버터를 엑세스하기 위한 CAN Access DLL에서 사용 가능한 기능들에 대하여 설명합니다.

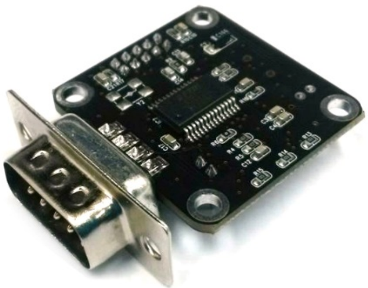


그림 1-1 USB2CAN

2 다운로드

2.1 Windows Desktop & Server

엔티렉스의 공식 홈페이지(ntrexgo.com)에서 LinKerS USB2CAN Library.zip 파일을 다운로드 받은 후 압축을 해제하면 CAN_Access.dll, CAN_Access.lib, CAN_Access.h, CAN_Test.exe 총 4개의 파일이 있는 것을 확인할 수 있습니다.

사용자는 별도의 개발 모듈을 사용하거나 또는, CAN_Test.exe를 통해서 USB2CAN을 테스트할 수 있습니다.

2.2 Max OS X, Linux, Windows CE

Mac OS X, Linux, Windows CE에서는 사용할 수 없습니다.

3 API Reference

3.1 CAN_OpenVcp

※ Summary

USB2CAN(UART) 컨버터를 엑세스하기 위해 핸들을 얻어옵니다.

※ Definition

CAN_HANDLE CAN_OpenVcp(const char *comPortName, long comBaudrate)

※ Parameters

comPortName : COM port 이름을 나타내는 문자열 포인터

comBaudrate : Baudrate 값을 나타내는 숫자: 921600bps로 고정

※ Return Value

성공한 경우 USB2CAN 컨버터의 핸들을 리턴하고 실패한 경우 -1을 리턴합니다.

※ Remarks

이 함수는 FTDI FT232칩을 기반으로 만들어진 USB2CAN 컨버터를 오픈하는데 사용합니다. FT232 칩의 Windows용 드라이버는 가상 시리얼 포트(VCP; Virtual COM Port) 드라이버를 제공하는데, 가상 시리얼 포트를 통해 COM 포트를 엑세스하는 것과 동일하게 USB2CAN 컨버터를 엑세스하게 됩니다.

※ Examples

```
CAN_HANDLE hBus = CAN_OpenVcp ("\\\\.\\COM3", 921600);
if (0 <= hBus) {
       CAN_SetTransferMode (hBus, 1);    // Open 성공
       }
else {
                                         // Open 실패, Error 처리
}
```

3.2 CAN_OpenUsb

※ Summary

USB2CAN(FIFO) 컨버터를 엑세스하기 위해 핸들을 얻어옵니다.

※ Definition

CAN_HANDLE CAN_OpenUsb(const char *serialNumber)

※ Parameters

serialNumber : USB2CAN의 Serial Number 문자열 포인터

※ Return Value

성공한 경우 핸들을 리턴하고 실패한 경우 -1을 리턴합니다.

※ Remarks

이 함수는 FTDI FT245칩을 기반으로 만들어진 USB2CAN 컨버터를 오픈 하는데 사용합니다. FT245 칩이 USB로 PC에 연결되었을 때 칩을 구분하는 Serial Number가 제공되는데, 이 Serial Number를 통해 FIFO 포트를 오픈하여 USB2CAN 컨버터를 엑세스하게 됩니다. Serial Number는 USB2CAN UI를 실행하면 다음과 같이 검색됩니다.

※ Examples

```
CAN_HANDLE hBus = CAN_OpenUsb ("NTWXHPOP");
if (0 <= hBus) {
      CAN_SetTransferMode (hBus, 1);    // Open 성공
}
else {
                                        // Open 실패, Error 처리
}
```

3.3 CAN_Close

※ Summary

USB2CAN 컨버터의 열린 핸들을 닫아 사용을 종료합니다.

※ Definition

int CAN_Close(CAN_HANDLE handle)

※ Parameters

handle : USB2CAN 컨버터의 핸들

※ Return Value

성공한 경우 1을 리턴하고 실패한 경우 0을 리턴합니다. 여기서 실패한 경우란, 열려있지 않은 핸들을 닫으려고 하는 경우입니다.

3.4 CAN_IsOpened

※ Summary

USB2CAN 핸들이 열려 있는지 확인합니다.

※ Definition

int CAN_IsOpened (CAN_HANDLE handle)

※ Parameters

handle : USB2CAN 컨버터의 핸들

※ Return Value

USB2CAN 컨버터의 핸들이 열려있는 경우 1을 리턴하고 열려있지 않은 경우 0을 리턴합니다.

3.5 CAN_ SetTimeout

※ Summary

읽기와 쓰기 요청에 대한 타임아웃 값을 밀리 초 단위로 설정합니다.

※ Definition

int CAN_SetTimeout (CAN_HANDLE handle, unsigned long readTimeout, unsigned long writeTimeout, unsigned long latencyTimer)

※ Parameters

handle : USB2CAN 컨버터의 핸들

readTimeout : 읽기 요청에 대한 타임아웃 값 (범위: 1 ~ 1000 ms)

writeTimeout : 쓰기 요청에 대한 타임아웃 값 (범위: 1 ~ 1000 ms)

latencyTimer : USB2CAN(UART)인 경우, Inter character timeout 값 (범위: 1~1000 ms);

USB2CAN(FIFO)인 경우, Latency timer 값 (범위: 0 ~ 32 ms)

※ Return Value

성공한 경우 1을 리턴하고 실패한 경우 0을 리턴합니다.

※ Remarks

Inter character timeout은 문자를 받은 후 다음 문자를 받기까지 얼마나 기다릴 것인지를 밀리 초 단위로 설정합니다.

Latency timer는 수신 버퍼에 문자가 들어왔는지를 체크하는 타이머 주기를 설정합니다. 이 값을 줄이면 수신 버퍼를 체크하는 주기가 짧아져 CPU의 부하를 증가시키고 읽기 요청에 대한 응답속도가 빨라집니다. 반대로 이 값을 늘이면 CPU의 부하를 줄이는 대신 읽기 요청에 대한 응답속도가 느려집니다.

※ Examples

```
CAN_HANDLE hBus = CAN_OpenUsb ("NTWXHPOP");
if (0 <= hBus) {
       CAN_SetTransferMode (hBus, 1);
       CAN_SetTimeout (hBus, 33, 100, 0);
              // Open 성공
}
else {
              // Open 실패, Error 처리
}
```

3.6 CAN_CountRxQueue

※ Summary

수신 버퍼에 들어온 문자 수를 셉니다.

※ Definition

int CAN_CountRxQueue (CAN_HANDLE handle)

※ Parameters

handle : USB2CAN 컨버터의 핸들

※ Return Value

성공한 경우 수신 버퍼의 문자 수를 리턴하고 실패한 경우 0을 리턴합니다. 즉, 0이 리턴된 경우는 함수 호출이 실패하였거나 수신 버퍼에 들어온 문자가 없다는 의미입니다.

※ Examples

```
while (1) {
            if (0 < CAN_CountRxQueue (hBus)) {
                    long ID;
                    int length;
                    char data[8];
                    int ext, rtr;
                    if (CAN_Recv (hBus, &ID, &length, data, &ext, &rtr)) {
                    // CAN message 읽기 성공
                    }
            }
            else {
                    Sleep (10);
}
}
```

3.7 CAN_ Purge

※ Summary

송신과 수신 버퍼의 문자들을 비웁니다.

※ Definition

void CAN_Purge (CAN_HANDLE handle)

※ Parameters

handle : USB2CAN 컨버터의 핸들

3.8 CAN_ Recv

※ Summary

수신 버퍼의 문자열로부터 하나의 CAN 메시지를 읽습니다.

※ Definition

int CAN_Recv (CAN_HANDLE handle, long *ID, int *length, char data[8], int *Ext, int *RTR)

※ Parameters

handle : USB2CAN 컨버터의 핸들

ID : CAN 메시지를 송신한 장치의 ID를 담는 변수의 포인터

(범위: Standard message인 경우 0x000 ~ 0x7FF,

Extended message인 경우 0x00000000 ~ 0x1FFFFFFF)

length : CAN 메시지에서 data의 길이를 담는 변수의 포인터 (범위: 0 ~ 8 byte)

data : CAN 메시지의 데이터를 저장하는 배열의 포인터 (크기: 최대 8 byte)

Ext : 수신된 CAN ID가 Standard혹은 Extended인지를 구분하는 변수의 포인터:

- 변수의 값이 0인 경우 Standard CAN message,
- 변수의 값이 1인 경우 Extended CAN message

RTR : 수신된 CAN 메시지의 종류가 RTR 인지를 구분하는 변수의 포인터:

- 변수의 값이 0인 경우 데이터 전송,
- 변수의 값이 1인 경우 RTR로 데이터 전송 요청

*** RTR(Remote Transmission Request)** - 일반적으로 데이터 전송은 센서와 같은 데이터를 발생시키는 노드에 의해 자동으로 Data Frame을 전송되게 됩니다. 어쨌거나, 데이터를 수집하는 노드가 Remote Frame을 보냄으로 데이터 전송을 요청하는 것도 가능합니다.

※ Return Value

CAN 메시지를 읽기 성공한 경우 1을 리턴하고 실패한 경우 0을 리턴합니다.

※ Remarks

CAN_Recv() 함수는 수신 버퍼에 수신된 문자열이 존재하고, 이 문자열로부터 하나의 CAN 메시지를 조립할 수 있다면 즉각적으로 리턴합니다. 하지만 수신 버퍼에 문자가 수신되지 않았거나 수신된 문자열이 하나의 CAN 메시지를 조립하기에 부족하다면 CAN_SetTimeout() 함수로 설정된 readTimeout 시간 동안 완전한 문자열이 수신되기를 기다립니다.

※ Examples

```
while (1) {
        long ID;
        int length;
        char data[8];
        int ext, rtr;
        if (CAN_Recv (hBus, &ID, &length, data, &ext, &rtr)) {
                        // CAN message 읽기 성공
        }
        else {
                Sleep (10);
        }
}
```

3.9 CAN_Send

※ Summary

CAN 메시지를 전송합니다.

※ Definition

int CAN_Send (CAN_HANDLE handle, long ID, int length, char data[8], int Ext, int RTR)

※ Parameters

Handle : USB2CAN 컨버터의 핸들

ID : CAN 메시지를 수신할 장치의 ID

(범위: Standard message인 경우 0x000 ~ 0x7FF,

Extended message인 경우 0x00000000 ~ 0x1FFFFFFF)

length : CAN 메시지에서 data의 길이를 저장(범위: 0 ~ 8 byte)

data : CAN 메시지의 데이터를 저장하는 배열의 포인터 (크기: 최대 8 byte)

Ext : 전송할 CAN ID가 Standard혹은 Extended인지를 구분하는 변수:

- 변수의 값이 0인 경우 Standard CAN message,
- 변수의 값이 1인 경우 Extended CAN message

RTR : 전송할 CAN 메시지의 종류가 RTR 인지를 구분하는 변수:

- 변수의 값이 0인 경우 데이터 전송,
- 변수의 값이 1인 경우 RTR로 데이터 전송 요청

※ Return Value

CAN 메시지의 전송을 성공한 경우 1을 리턴하고 실패한 경우 0을 리턴합니다.

※ Examples

```
char data[8] = {0x83, 0x11, 0x00, 0x03, };      // 최대8 byte

if (CAN_Send (hBus, 0x11, 4, data, 0, 0)) {
                                                // 전송성공
}
```

3.10 CAN_ GetConfig

※ Summary

CAN 포트의 설정을 읽어옵니다.

※ Definition

int CAN_GetConfig (CAN_HANDLE handle, long *bitrate, unsigned long *filterID, unsigned long *filterMask)

※ Parameters

handle : USB2CAN 컨버터의 핸들

bitrate : CAN 통신속도를 저장하는 변수의 포인터 (단위: kBps)

filterID : CAN Filter Identifier를 담는 변수의 포인터 (범위: 0x0 ~ 0x1FFFFFFF)

filterMask : CAN Filter Mask를 담는 변수의 포인터 (범위: 0x0 ~ 0x1FFFFFFF)

※ Return Value

성공한 경우 1을 리턴하고 실패한 경우 0을 리턴합니다.

※ Examples

```
long bitrate;
unsigned long filterID;
unsigned long filterMask;

if (CAN_GetConfig (hBus, &bitrate, &filterID, &filterMask)) {
                        // 설정값 읽기 성공
}
```

3.11 CAN_ SetConfig

※ Summary

CAN 포트의 설정을 USB2CAN 컨버터로 내려 보냅니다.

※ Definition

int CAN_SetConfig (CAN_HANDLE handle, long bitrate, unsigned long filter_id, unsigned long filter_mask)

※ Parameters

handle : USB2CAN 컨버터의 핸들

bitrate : CAN 통신속도 (단위: kBps).

설정 가능한 CAN 통신 속도는 다음과 같습니다:

- 1000 - 1M Bps
- 800 - 800K Bps
- 500 - 500K Bps
- 250 - 250K Bps
- 125 - 125K Bps
- 50 - 50K Bps
- 25 - 25K Bps
- 10 - 10K Bps

filterID : CAN Filter Identifier (범위: 0x0 ~ 0x1FFFFFFF)

filterMask : CAN Filter Mask (범위: 0x0 ~ 0x1FFFFFFF)

※ Return Value

성공한 경우 1을 리턴하고 실패한 경우 0을 리턴합니다.

※ Remarks

CAN_SetConfig() 함수가 호출되면 USB2CAN 컨버터에서는 다음과 같은 과정을 거치게 됩니다:

① bitrate, filterID, filterMask 값으로 CAN port를 설정합니다.

② bitrate, filterID, filterMask 값을 Flash memory에 저장합니다.

이 함수에 의해 설정된 값은 USB2CAN 컨버터가 재시작 되어도 Flash memory에 저장된 설정값에 의해 CAN port의 설정이 유지됩니다.

※ Examples

```
if (CAN_GetConfig (hBus, 1000, 0x0000, 0x0000)) {
            // 설정값 읽기 성공
}
```

3.12 CAN_ SetTransferMode

※ Summary

컨버터의 CAN port와 PC간 패킷 전송을 활성화/비활성화 하거나 전송 방식을 결정합니다. 선택 가능한 전송 방식은 다음과 같습니다.

0 – CAN port와 PC간 CAN 메시지 패킷만의 송수신을 금지합니다. 하지만 설정이나 명령과 관련된 패킷의 송수신은 가능합니다.

1 – USB2CAN 컨버터와 PC간 바이너리 형식의 패킷을 송수신 합니다. 이 때, CAN 메시지 및 설정, 명령과 관련된 패킷의 송수신 형태는 바이너리 형식으로 인코딩 됩니다.

2 - USB2CAN 컨버터와 PC간 텍스트 형식의 패킷을 송수신 합니다. 이 때, CAN 메시지 및 설정, 명령과 관련된 패킷의 송수신 형태는 텍스트 형식으로 인코딩 됩니다.

※ Definition

int CAN_SetTransferMode (CAN_HANDLE handle, int mode)

※ Parameters

handle : USB2CAN 컨버터의 핸들

mode : 패킷 전송 모드 설정:

- 0 – Disable transmit
- 1 – 패킷을 Binary 형식으로 인코딩 하여 송 수신
- 2 - 패킷을 Text 형식으로 인코딩 하여 송 수신

※ Return Value

성공한 경우 1을 리턴하고 실패한 경우 0을 리턴합니다.

※ Remarks

CAN_Access.DLL 파일에서는 Text/Binary 형식의 인코딩이 내부적으로 처리되기 때문에 사용자에게 Text와 Binary 형식의 전송 방식을 구분하는 것은 중요하지 않습니다. 그 이유는 사용자는 CAN_Recv(), CAN_Send() 함수에 의해 항상 디코딩 된 CAN message 형식으로 읽고 쓰기 때문입니다.

또한, 전송 모드의 설정과 상관없이, PC에서 USB2CAN 컨버터로 CAN 메시지를 전송할 때는 항상 바이너리 형식으로 인코딩 된 패킷을 전송하게 됩니다. 그렇기 때문에, 사용자가 텍스트 모드로 변경하더라도 패킷을 주고받는 과정에서 바이너리 형식으로 변경하게 됩니다.

관련 자료

아래 홈페이지에서 LinKerS 관련 문서와 자료를 확인하실 수 있습니다.

- 엔티렉스:
  http://www.ntrexgo.com/
- 디바이스마트:
  http://www.devicemart.co.kr/
- MoonWalker USB2CAN (FIFO) 판매페이지:
  http://www.devicemart.co.kr/goods/view.php?seq=1111719
- MoonWalker USB2CAN (VCP) 판매페이지:
  http://www.devicemart.co.kr/goods/view.php?seq=1111711

- MoonWalker 소개:
  http://www.ntrexgo.com/moonwalker-%EC%86%8C%EA%B0%9C
- MoonWalker 판매페이지:
  http://devicemart.co.kr/goods/brand.php?seq=1475
- MoonWalekr UI Utility:
  http://www.ntrexgo.com/archives/19482
- MoonWalker 자료:
  http://www.ntrexgo.com/archives/category/ntrex-lab/moonwalker
- MoonWalker 예제:
  http://www.ntrexgo.com/archives/category/ntrex-lab/moonwalker_applications
- MoonWalker 액세서리:
  http://www.ntrexgo.com/archives/category/moonwalker_accessory

문서 변경 이력

Data	Version	Charges
2014.03.25	1.00	- 첫 출시