Spresense用CAN AddOnボード

更新履歴

版数	更新内容	更新日
0.2	Refsを追加した。	2023/01/12
0.1	初版を0.1版とする。	2022/10/04

目次

CAN 概要

- **Controller Area Network (CAN)** は、ビークルバス規格の一種で、ホストコンピュータなしでマイクロコントローラやデバイスが相互に通信できるように設計されている。耐ノイズ性の強化が考慮された堅牢な規格である。メッセージベースのプロトコルであり、元々は、自動車内部の多重化電気配線用に設計されたものだが、機器の制御情報の転送用として普及しており、輸送用機械、工場、工作機械などのロボット分野においても利用されている。自動車においては、速度、エンジンの回転数、ブレーキの状態、故障診断の情報などの転送に使用されている。
- データ転送速度は、40mの通信路においては最高で1Mbps程度、500mの通信路においては最高で125kbps程度。実際の運用においては、速いもので500kbps、遅いもので83.3kbpsとなっている。通信速度が上がるほど接続できる機器の数が減るので、高級車などでは速度の異なる複数の通信路をもつ。

- **CAN メッセージ**
  1. CANメッセージは多くのセグメントを含んでいます。しかし、2つの主なセグメントは識別子とデータです。
  2. 識別子はCANネットワークでCANデバイスを識別することに使用されています。そして、データは1つのデバイスから別のデバイスへ制御データとなります。
  3. 識別子の長さは11または29ビットです。
     1. 標準CAN識別子 は11 bitです。
     2. 拡張CAN識別子 は29 bitです。
  4. データは0から8バイトまでです。
- **CAN BUSプロトコルを使う理由**
  1. 低コスト
  2. 集中型
  3. 柔軟性がある
  4. ロバスト
  5. 効率的

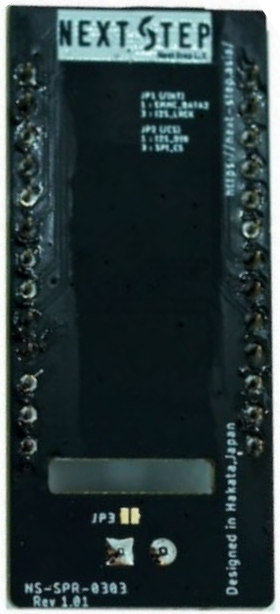
詳しくは下記をご参照ください。

[Refs]

https://ja.wikipedia.org/wiki/Controller_Area_Network

https://jp.seeedstudio.com/blog/2020/08/05/introduction-to-can-bus-and-how-to-use-it-with-arduino/

Spresense 用 CAN AddOnボード概要



Spresense 用のCAN通信ボードです。CANは車載ECUでは標準で使われている通信規格で自動車から色々なデータを取り出せます。CANチップはMicrochip社のMCP2515T-E_ST、CANドライバーは3.3Vで動作するInfineon社のTLE7251VSJXUMA1を使用しています。

詳しくは下記をご参照ください。
[Refs]
https://nextstep.official.ec/items/38434337

Spresense 用 CAN AddOnボード事前はんだ付けは必須

事前準備
下記3か所はんだ付けして、ショットさせる必要です。

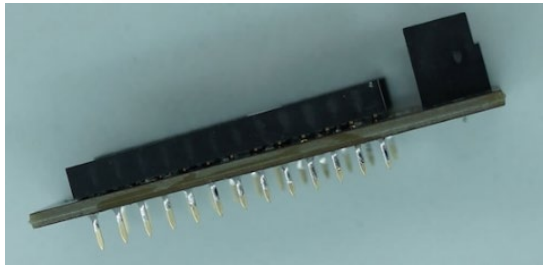
はんだ付け情報は下記を参照しました。
[Refs]
https://qiita.com/kaz19610303/items/c43bdd8134b3a47b0e3e

Spresenseメインボード + CAN AddOnボード 物理接続図

Spresenseメインボード

CAN AddOnボード**表面図**

CAN AddOnボード**側面図**



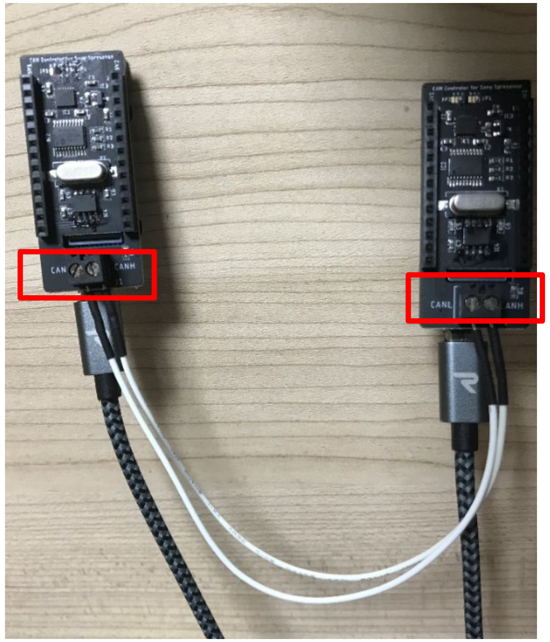
CAN_loopbackサンプルの動作確認

CAN接続なし、
単体で送受信の
動作確認できる

[log]
Entering Configuration Mode Successful!
Setting Baudrate Successful!
MCP2515 Initialized Successfully!
MCP2515 Library Loopback Example...
Message Sent Successfully!
Standard ID: 0x100 DLC: 8 Data: 0xAA 0x55 0x01 0x10 0xFF 0x12 0x34 0x56
Message Sent Successfully!
Standard ID: 0x100 DLC: 8 Data: 0xAA 0x55 0x01 0x10 0xFF 0x12 0x34 0x56
Message Sent Successfully!
Standard ID: 0x100 DLC: 8 Data: 0xAA 0x55 0x01 0x10 0xFF 0x12 0x34 0x56

[サンプル]
https://github.com/SonySemiconductorSolutions/ssup-spresense/tree/main/Arduino/MCP2515T-E_ST/CAN_loopback
[Refs]
https://qiita.com/kaz19610303/items/c43bdd8134b3a47b0e3e
https://github.com/coryjfowler/MCP_CAN_lib

CAN送受信サンプルの動作確認



接続
CANL⇔CANL
CANH⇔CANH

２つの[Spresense 用 CAN AddOnボード]を使用します。1つは送信側として、8byteのデータを1秒間隔にCANで送信し続きます。もう1つは受信側として、受信したデータを表示し続きます。

送信成功ログ

```
Entering Configuration Mode Successful!
Setting Baudrate Successful!
MCP2515 Initialized Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
```

受信成功ログ

```
[log]
Entering Configuration Mode Successful!
Setting Baudrate Successful!
MCP2515 Initialized Successfully!
MCP2515 Library Receive Example...
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
```

[サンプル]
https://github.com/SonySemiconductorSolutions/ssup-spresense/tree/main/Arduino/MCP2515T-E_ST/CAN_send
[Refs]
https://qiita.com/kaz19610303/items/c43bdd8134b3a47b0e3e
https://github.com/coryjfowler/MCP_CAN_lib

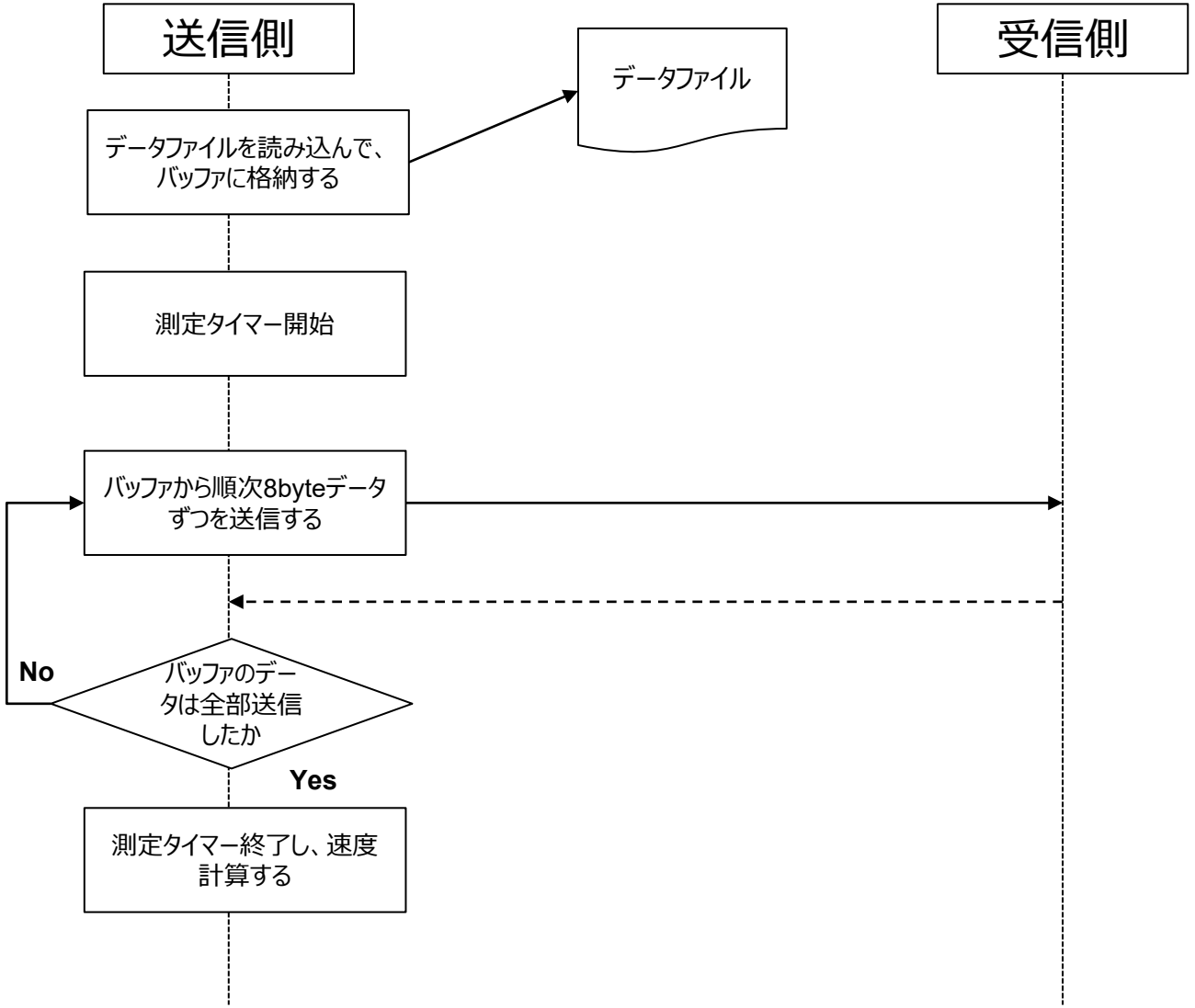
CAN送信速度測定方法

【サンプル概要】

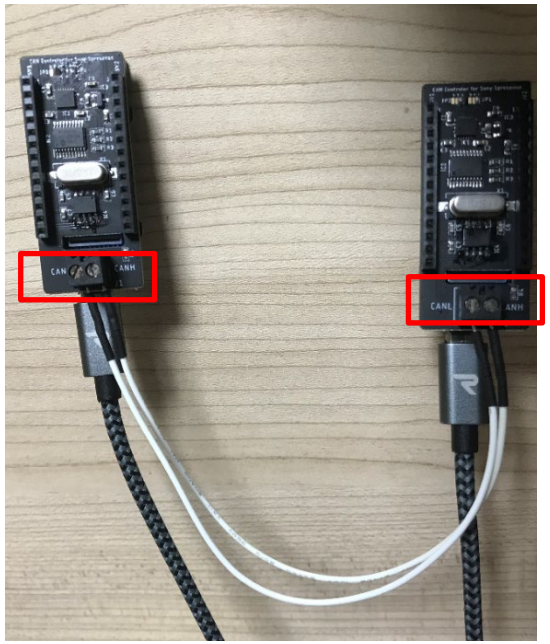
本サンプルはSpresenseのフラシュメモリもしくはSDカードからデータファイルを読み込んで、バッファに設定します。そのバッファから8byteずつ分けてデータファイルサイズまで送信し続きます。そして、繰り返して10回速度を測定を行い、平均値を計算して出力します。

【注意】

バッファサイズは200Kbyteを設定しているため、データファイルのサイズをそれ以下に抑える必要です。



CAN送受信速度測定サンプルの動作確認



接続
CANL⇔CANL
CANH⇔CANH

受信速度は送信速度と同じであることを確認しました。

※受信処理はログ出力などの処理があるため、時間がかかります。正確な受信速度測定するために、受信した後のログ出力などの処理をコメントアウトする必要です。

[log]
Entering Configuration Mode Successful!
Setting Baudrate Successful!
MCP2515 Initialized Successfully!
t_len = 153600

Count 1: Sent 1228800 bits in 13.9441 seconds, rate = 86.06 Kbits/second
Count 2: Sent 1228800 bits in 13.9441 seconds, rate = 86.06 Kbits/second
Count 3: Sent 1228800 bits in 13.9441 seconds, rate = 86.06 Kbits/second
Count 4: Sent 1228800 bits in 13.9442 seconds, rate = 86.06 Kbits/second
Count 5: Sent 1228800 bits in 13.9442 seconds, rate = 86.06 Kbits/second
Count 6: Sent 1228800 bits in 13.9442 seconds, rate = 86.06 Kbits/second
Count 7: Sent 1228800 bits in 13.9442 seconds, rate = 86.06 Kbits/second
Count 8: Sent 1228800 bits in 13.9441 seconds, rate = 86.06 Kbits/second
Count 9: Sent 1228800 bits in 13.9444 seconds, rate = 86.06 Kbits/second
Count 10: Sent 1228800 bits in 13.9443 seconds, rate = 86.06 Kbits/second
86.06 86.06 86.06 86.06 86.06 86.06 86.06 86.06 86.06 86.06
average rate:86.06Kbits/second

[サンプル]
https://github.com/SonySemiconductorSolutions/ssup-spresense/tree/main/Arduino/MCP2515T-E_ST/CAN_send_speed

Appendix

標準CAN、拡張CANについて

標準CANの場合、識別子は11bitです。拡張CANの場合、識別子は29bitです。
標準CANと拡張CANの使い分けは下記のsendMsgBuf APIのextパラメータで決まります。
extが0の場合、標準CANを使います。
extが1の場合、拡張CANを使います。

```
INT8U MCP_CAN::sendMsgBuf(INT32U id, INT8U ext, INT8U len, INT8U *buf)
標準CAN送信使用例：byte sndStat = CAN0.sendMsgBuf(0x100, 0, 8, data);
拡張CAN送信使用例：byte sndStat = CAN0.sendMsgBuf(0x100, 1, 8, data);
```

標準CANと拡張CANは混在して使っても問題がありません。

２つのSpresense 用 CAN AddOnボードを使用します。1つは送信側として、8byteのデータを1秒間隔に標準CANと拡張CANでクロスで送信し続きます。もう1つは受信側として、受信したデータを表示し続きます。

※送信側のサンプルは下記となります。受信側は変更がないです。

```
void loop()
{
  // send data:  ID = 0x100, Standard CAN Frame, Data length = 8 bytes, 'data' = array of data bytes to send
  byte sndStat;
  bool isIDStandard = true;
  if ( isIDStandard ) {
    sndStat = CAN0.sendMsgBuf(0x100, 0, 8, data);
  } else {
    sndStat = CAN0.sendMsgBuf(0x1000, 1, 8, data);
  }
  if(sndStat == CAN_OK){
    Serial.println("Message Sent Successfully!");
  } else {
    Serial.println("Error Sending Message...");
  }
  isIDStandard = !isIDStandard;
  delay(INTERVEL);   // send data per 100ms
}
```

送信成功ログ

```
Entering Configuration Mode Successful!
Setting Baudrate Successful!
MCP2515 Initialized Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
Message Sent Successfully!
```

受信成功ログ

```
Entering Configuration Mode Successful!
Setting Baudrate Successful!
MCP2515 Initialized Successfully!
MCP2515 Library Receive Example...
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Extended ID: 0x00001000  DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Extended ID: 0x00001000  DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Standard ID: 0x100       DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
Extended ID: 0x00001000  DLC: 8  Data: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
```

[サンプル]
https://github.com/SonySemiconductorSolutions/ssup-spresense/tree/main/Arduino/MCP2515T-E_ST/CAN_send_std_ext

CAN送信エラーについて

```
void setup()
{
  Serial.begin(115200);

  // Initialize MCP2515 running at 16MHz with a baudrate of 500kb/s and the masks and filters disabled.
  if(CAN0.begin(MCP_ANY, CAN_500KBPS, MCP_16MHZ) == CAN_OK) {
    Serial.println("MCP2515 Initialized Successfully!");
  }
  else {
    Serial.println("Error Initializing MCP2515...");
  }

  CAN0.setMode(MCP_NORMAL);   // Change to normal mode to allow messages to be transmitted
}
```

CAN0.begin(MCP_ANY, CAN_500KBPS, MCP_16MHZ)を下記のようにbaudrateと周波数を増やして確認したら、右記のようにエラーが発生します。

```
CAN0.begin(MCP_ANY, CAN_1000KBPS, MCP_16MHZ)
CAN0.begin(MCP_ANY, CAN_1000KBPS, MCP_20MHZ)
CAN0.begin(MCP_ANY, CAN_500KBPS, MCP_20MHZ)
```

送信エラー原因

送受信の初期化処理のところに、CAN速度とMCP周波数は同じで設定しなければならない。
送信側は初期化するときに、下記のように送信速度を設定しましたが、

```
CAN0.begin(MCP_ANY, CAN_1000KBPS, MCP_16MHZ)
```

受信側は下記のままで、送信側と同じの速度を設定しませんでした。

```
CAN0.begin(MCP_ANY, CAN_500KBPS, MCP_16MHZ)
```

[Refs]

https://qiita.com/kaz19610303/items/c43bdd8134b3a47b0e3e

https://github.com/coryjfowler/MCP_CAN_lib

```
#define CAN_GETTXBFTIMEOUT (6)
#define CAN_SENDMSGTIMEOUT (7)
```

```
[log]
Entering Configuration Mode Successful!
Setting Baudrate Successful!
MCP2515 Initialized Successfully!
t_len = 153600
sndStat = 7
Error Sending Message...
sndStat = 7
Error Sending Message...
sndStat = 7
Error Sending Message...
sndStat = 6
Error Sending Message...
sndStat = 6
Error Sending Message...
sndStat = 6
Error Sending Message...
sndStat = 6
```

CAN実効送信速度について

右記の表により、送信側と受信側を同じのCAN速度とMCP周波数の組み合わせパターンで実効送信速度(単位：Kbps)測定しました。

【パラメータ設定】

MCP周波数は同じの場合

CAN速度が大きく設定すればするほど、実効送信速度は大きくなる。

CAN速度は同じの場合

MCP周波数が小さく設定すればするほど、実効送信速度は大きくなる。

【エラー】

Error1の場合

初期設定失敗して、送信できない。

Error2の場合

詳細は次のページをご参照ください。

CAN速度(Kbps)	MCP周波数(Mhz)		
	8	16	20
CAN_4K096BPS	Error1	Error1	Error1
CAN_5KBPS	Error2	Error2	Error1
CAN_10KBPS	Error2	Error2	Error1
CAN_20KBPS	Error2	Error2	Error1
CAN_31K25BPS	Error2	Error1	Error1
CAN_33K3BPS	Error2	Error2	Error1
CAN_40KBPS	Error2	Error2	Error2
CAN_50KBPS	Error2	75.01	Error2
CAN_80KBPS	36.15	Error2	Error2
CAN_100KBPS	43.69	Error2	Error2
CAN_125KBPS	52.3	Error2	Error2
CAN_200KBPS	75.01	43.69	36.15
CAN_250KBPS	86.05	52.3	43.69
CAN_500KBPS	126.52	86.05	75.01
CAN_1000KBPS	173.03	126.52	116.94

```
Error1:Setting Baudrate Failure...
Error2:SENDMSGTIMEOUT error
```

CAN実効送信速度について

【テスト要素】

送信トータルデータサイズ：28672 byte

固定一回送信パケットデータサイズ：8 byte

送信回数： 28672 / 8 = 3584

【Error2について】

Error2は送信するときに発生したSENDMSGTIMEOUTエラーです。また右記すべてのパターンでError2発生した回数は全部3584です。送信は一回も成功していないように見えます。

しかし、受信側で確認すると、データ受信があるので、右記の青字はそれぞれのパターンでデータ受信できた回数を表しています。

CAN速度(Kbps)	MCP周波数(Mhz)		
	8	16	20
CAN_4K096BPS	Error1	Error1	Error1
CAN_5KBPS	Error2:719	Error2:361	Error1
CAN_10KBPS	Error2:1197	Error2:719	Error1
CAN_20KBPS	Error2:3584	Error2:1197	Error1
CAN_31K25BPS	Error2:3584	Error1	Error1
CAN_33K3BPS	Error2:3584	Error2:1794	Error1
CAN_40KBPS	Error2:3584	Error2:3584	Error2:1794
CAN_50KBPS	Error2:3584	75.01	Error2:3584
CAN_80KBPS	36.15	Error2:3584	Error2:3584
CAN_100KBPS	43.69	Error2:3584	Error2:3584
CAN_125KBPS	52.3	Error2:3584	Error2:3584
CAN_200KBPS	75.01	43.69	36.15
CAN_250KBPS	86.05	52.3	43.69
CAN_500KBPS	126.52	86.05	75.01
CAN_1000KBPS	173.03	126.52	116.94

```
Error1:Setting Baudrate Failure...
Error2:SENDMSGTIMEOUT error
```