HRP CAN Interface Board

SIF-0069

ソフトウェアマニュアル（第4版）

2008年 8 月

目　次

1. 概要

本マニュアルは HRP CAN Interface Board[SIF-0069](以下、本ボード)が持つ機能をユーザが簡単に扱うことができるよう製作されたデバイスドライバと、そのドライバにアクセスする為の手順などについて説明します。SIF-0069 のハードウェア詳細については取扱説明書を参照ください。

デバイスドライバはソース形式で配布いたしますのでコンパイルする必要があります。

関数は各機能別に用意され各形式に沿ったデータを入力することで必要なデータを取得または出力することができます。

2. 動作環境

動作環境を以下に示します。

表1. 動作環境

項	項　　目	仕　　様	
1	対応ボード	HRP CAN Interface Board(PCI-104 ボード) SIF-0069	
2	OS	Linux kernel ver2.6/2.4	
3	ドライバ配布形式	ソース(makefile 附属)	
4	ドライバモジュールの種類	キャラクタ型	
5	動作確認済み ディストリビューションと コンパイル環境	VineLinux4.2[2.6.16]	gcc 3.3.6 make 3.80
		VineLinux3.2[2.4.31]	gcc 3.3.2 make 3.80

3. ファイル構成

デバイスドライバのファイル構成を示します。

zcan/driver/		
	Zc_IOCTL.c	I/O コントロール
	Zc_Func.h	各関数の宣言
	Zc_Func.c	ヘッダーの宣言
	Zc_IoFunc.c	各関数のソース
	Zc_Base.h	PCI 情報ヘッダ
	Zc_Util.h	補助用マクロヘッダ
	Makefile	メイクファイル（2.4/2.6 対応）
	(zcan.o)	デバイスドライバモジュール／カーネルバージョン 2.4 用
	(zcan.ko)	デバイスドライバモジュール／カーネルバージョン 2.6 用
zcan/lib/		
	Zc_IO.c	ユーザ関数モジュールソース
	Zc_IO.h	ユーザ関数モジュールヘッダ
	(Zc_IO.o)	ユーザ関数モジュール
zcan/sample/		
	Sample.c	サンプルソース
	Makefile	メイクファイル
	(Sample)	サンプル実行プログラム

※()内のファイル名は make で作成されるファイルです。

4. インストール手順

本ソフトウェアは全てスーパーユーザで動作致します。root でログインするか「su」コマンドでスーパーユーザになりインストールやプログラムの作成・実行を行ってください。

1) ファイルの準備

○ホームページからのダウンロード（現在準備中）

弊社ウェブページの「ダウンロード」から「zcan-X.X.X.tar.gz」のファイルをダウンロードしてください。（X はバージョン番号）

○フロッピーから取得

ドライブに挿入し以下コマンドでマウントします。

#mount -t vfat /dev/fd0 /mnt/floppy

マウントしたフォルダに「zcan-X.X.X.tar.gz」がありますので適当な場所に移動してください。

#cp /mnt/floppy/zcan-X.X.X.tar.gz ./

※フロッピーは Windows のファイル形式でフォーマットされていますので「-t vfat」のオプションをつけてください。

2) ファイルの解凍・展開

「zcan-X.X.X.tar.gz」ファイルを適当なディレクトリに移動させ（ここでは/root）解凍・展開します。

#tar xvzf zcan-X.X.X.tar.gz

解凍・展開されたファイルが**3. ファイル構成**に記載されたファイルと同じであることを確認します。

3) ドライバのコンパイル

./zcan/driver へ移動します。

#cd /root/zcan/driver

「make」コマンドでコンパイルを始めます。

#make

コンパイルが終了するとカーネルバージョンが 2.6.x は「zcan.ko」、カーネルバージョンが 2.4.x の時は「zcan.o」というドライバモジュールができます。

※再度 make する場合は「make clean」を実行してください。

4) デバイスの登録・ドライバモジュールのインストール

下記コマンドを実行します。

#make dev

#make insmod

ドライバモジュールが正しくインストールされているか確認するには

#lsmod

と入力し表示されている文字の中に「zcan」が表示されれば問題ありません。

5) モジュールのアンインストール

#make rmmod

※ドライバモジュールをインストールできない場合は、メジャー番号を変更してみてください。変更個所は

Zc_Base.h の 63 行目　　#define　ZCAN_MAJOR　　**99**

Makefile の 10 行目　　major := **99**

メジャー番号は使われていない番号を選んで変更するのですが、その番号を探すには「ls -l /dev」で表示されるカンマで区切られた2つの数字の内、左側(1, 10, 4)がメジャー番号になりますので、その中からお探しください。(「grep」等使うと楽に探し出せます。)

```
#ls -l /dev
crw-rw-rw- 1 root  root      1, 3   Feb 23 1999     null
crw------- 1 root  root     10, 3   Feb 23 1999     psaux
crw------- 1 rubini tty      4, 3   Aug 16 22:22    tty1
```

変更後は必ず 3)の手順に戻ってください。

5. サンプルプログラム

各機能の基本的なコードの記述方法を書いたサンプルプログラムを用意しています。sample ディレクトリにある Makefile を利用して make すると「Sample」という実行ファイル作成されます。

#./Sample

make 時に関数を使用できるモジュール「Zc_IO.o」も作成されますのでこれを利用してオリジナルのプログラムを作成してください。

※ドライバモジュールのインストール前に Sample を実行した場合、ドライバモジュールをインストールできなくなります。再度インストールするには再起動が必要なので注意してください。

6. CAN について

本ボードは ISO 11898-1, CAN 2.0A, と CAN 2.0B 規格に準拠したボードですので基本的な動作に関しては仕様等を参考にしてください。この章では、基本仕様の補足と本ボードが持つ機能について説明します。

6.1. モード

本ボードはコンフィグレーション・通常動作・ループバック・スリープの4つのモードを持っています。各モードについて説明します。

1) コンフィグレーションモード

本モードは各種設定を行うときに使用します。

他のどのモードからでも本モードへ移行することができます。また、リセットがかかった時（リセット関数を呼び出すなど）は、必ずこのモードから始まることになります。

モード中は送受信ができませんが、送受信 FIFO や送信 HPB のバッファへの読書きはできます。本モード前の送受信中のデータは保持されます。

移行できるモードは通常動作・スリープ・ループバックの3つです。

2)　通常動作モード

本モードは CAN バスへの通信を行うためのものです。

モード中は送受信ができるようになります。

移行できるモードはコンフィグレーション・スリープの2つです。

3)　ループバック

本モードは送受信の診断目的に使用します。

同じチャンネル内の送信・受信ラインが接続されるので、送信したデータは同じチャンネルの受信バッファに格納されます。

モード中は外部への送受信は行われなくなります。

移行できるモードはコンフィグレーションのみとなります。

4)　スリープ

本モードは通信状態を一時的に停止する時に使います。

送信バッファが空か、CAN バスがアイドル状態の場合、本モードに移行することができます。モード中に送信・受信が起きると直ちに本モードから通常動作に移行することになります。

移行できるモードはコンフィグレーションと通常動作の2つになります。

6.2.　ボーレートの説明

ボーレートの設定には**8.6.各種設定／戻り値コード**で定義されたコードを利用する方法とユーザが自分で設定する方法の2つがあります。

前者は通信コンフィグを設定する時に ZCAN_PARA_CONFIG のメンバである BaudRate にコードを指定することで必要なパラメータ(ZCAN_PARA_CONFIG のメンバ Prescaler, SyncJumpWidth, TimeSegment1,2)が自動的に入力され設定を行います。

例. ボーレートコードで 1Mbps を設定

```
ZCAN_PARA_CONFIG paraConfig;
paraConfig.BaudRate = CAN_BR_1M;
```

後者は通信コンフィグを設定する時に4つのパラメータ(Prescaler, SyncJumpWidth, TimeSegment1,2)にそれぞれ指定された範囲の値を入力することで設定を行います。但し、その時 BaudRate には CAN_BR_NULL を入力することを忘れないでください。

例. ボーレートを自分で 1Mbps に設定

```
ZCAN_PARA_CONFIG paraConfig;
paraConfig.BaudRate       = CAN_BR_NULL;
paraConfig.Prescaler      = 1;
paraConfig.SyncJumpWidth = 1;
paraConfig.TimeSegment1   = 6;
paraConfig.TimeSegment2   = 3;
```

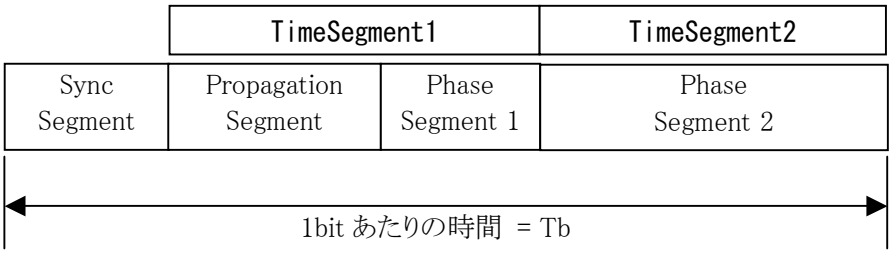
6.3. ボーレートのユーザ設定

ユーザが自分でボーレートを決める手順について説明します。

ボーレートや送受信には基準となるクロック(周期 Tq)が必要になるので、CAN に接続されたクロック(24MHz)から分周した Tq を求めます。

Tq = (Prescaler + 1) / 24 [us]

ボーレートを求めるのに必要な「1bit あたりの時間(Tb)」は CAN の仕様から下記図 1. のような関係にあります。



※SyncSegment は「1」に固定されていますのでプログラムで設定することはありません。

図1. bit タイミング

図1. の中にはいくつか項目がありますが、実際の設定では TimeSegment1,2 を使用しますので下記計算式で「1bit あたりの時間(Tb)」を求めることが出来ます。

Tb = Tq ×（1 ＋（TimeSegment1 ＋ 1）＋（ TimeSegement2 ＋ 1 ））

また、同期ずれ補償時間(Tsjw)を決める為に SyncJumpWidth の設定も行ってください。

Tsjw = Tq ×（SyncJumpWidth ＋ 1）

6.4. 追加機能についての説明

本ボードは1チャンネルあたり 64 メッセージの FIFO バッファが送受信ともに用意されています。(メッセージ･･･ID とデータ数、データを一塊にまとめた単位)

また送信には FIFO とは別に1メッセージの HPB(最優先バッファ)が用意されています。HPB は FIFO にたまっているバッファに関係なく最優先で送信を行います。ただし送信を中断して HPB を送信することはできません。送信前に TxBufType に設定することで送信バッファを選択することができます。

例. 送信バッファを FIFO バッファに設定

```
ZCAN_PARA_MESSAGE paraMessage;
paraMessage.TxBufType = CAN_TX_FIFO;
```

送信や FIFO のバッファの状態を確認するには Zc_GetStatus 関数を利用するとその状態を取得できます。受信の状態は Zc_GetStatus 関数で取得することができませんので、直接 Zc_GetMessage 関数でデータを読みに行きその戻り値で確認するか、割り込み状態を確認できる Zc_GetIntStatus 関数を利用してください。

Zc_GetIntStatus 関数は割込み許可が禁止されていても動作しますが、いったん確認したい状態フラグが立つとクリアしなければ変化しないままになります。再度 Zc_GetIntStatus 関数を使うには Zc_GetIntClear 関数で確認したい状態をクリアしてください。

例. 割り込み状態から受信の確認

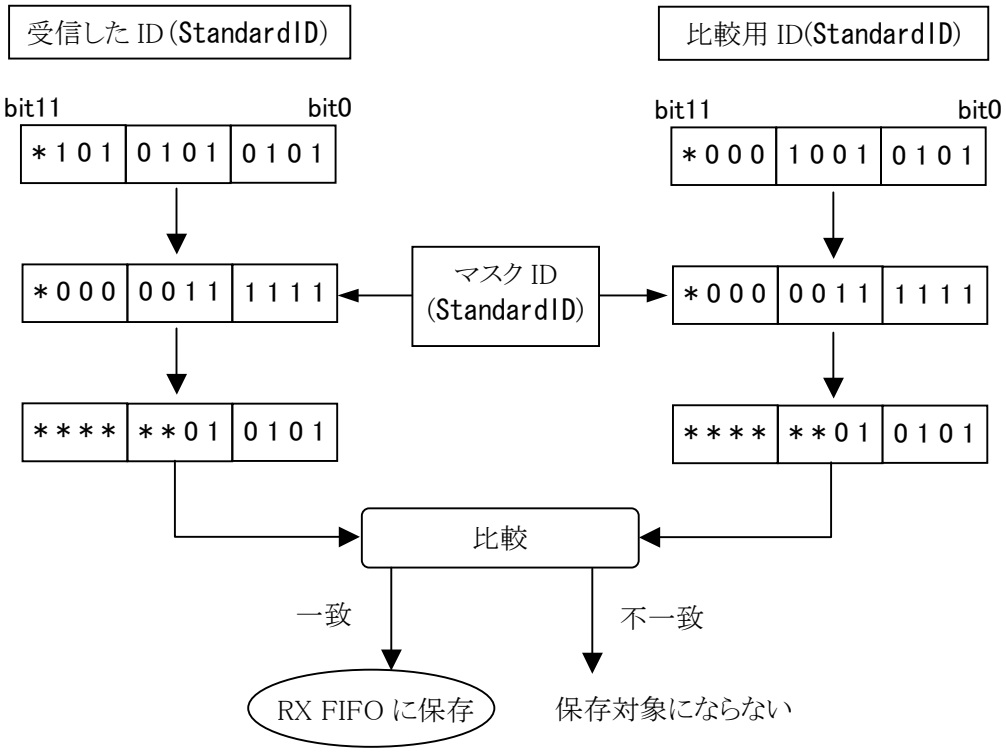
```
ZCAN_PARA_INTERRUPT paraInterrupt;
paraInterrupt.no = 0;
paraInterrupt.ch = CAN_CH1;
ZcanCAN_GetIntStatus(&paraInterrupt);
if( paraInterrupt.Condition & CAN_INT_RXOK ) printf( “Receive New Message.” );
```

6.5. アクセプタンスフィルタの使い方

本ボードには特定の CAN 識別子(StandardID、ExtendedID、IsExtended、FrameType をまとめたもの。以下 ID とします。)の受信データだけ取得できるアクセプタンスフィルタが実装されています。

アクセプタンスフィルタは CAN の各チャンネルに4つまで設定することができ、1つに付き2種類の設定用 ID があります。設定用 ID には受信した ID と比較する為の比較用 ID と、ID の各データをビット単位で比較に使用するかどうか決めるマスク ID があります。

アクセプタンスフィルタの一連の動作を図で説明します。(ここでは ID の StandardID のみを対象とします。)



※「＊」・・・比較対象外ビット

図2. アクセプタンスフィルタの動作

図 2 は StandardID のみですが、ID 内の全てのデータが比較され一致すれば RX FIFO に保存されます。一致しない場合は保存対象になりません。また、4 つあるアクセプタンスフィルタのいずれかに一致すれば RX FIFO に保存されることになります。

プログラムでは比較用 ID「ID」には比較したいデータを、マスク ID「Mask」にはマスクしたいデータを入力してください。マスク ID は受信した ID と比較用 ID のデータが重なるビットを「1」にすることで比較対象に、「0」にすることで比較対象外にすることができます。

また設定時には IsEnable にアクセプタンスフィルタを有効／無効にするのを忘れないでください。

例. アクセプタンスフィルタの設定

```
ZCAN_PARA_ACFILTER paraACFilter;
int ret;
paraACFilter.no = 0;
paraACFilter.ch = CAN_CH1;
paraACFilter.FilterNo = CAN_ACF_NO1;     // CAN_ACF_NO1～4 まで
paraACFilter.IsEnable = CAN_ACF_ENABLE;
paraACFilter.Mask.StandardID = 0x03F;    // 0000-0011-1111 (11bit)
paraACFilter.Mask.ExtendedID = 0x000FF;  // 0000-0000-0000-1111-1111 (18bit)
paraACFilter.Mask.IsExtended = CAN_MASK_ENABLE;  // 1
paraACFilter.Mask.FrameType  = CAN_MASK_DISABLE; // 0
paraACFilter.ID.StandardID   = 0x595;    // 0101-1001-0101 (11bit)
paraACFilter.ID.ExtendedID   = 0x25555;  // 0010-0101-0101-0101-0101 (18bit)
paraACFilter.ID.IsExtended   = CAN_EID_ENABLE;
paraACFilter.ID.FrameType    = CAN_FRAME_DATA;
ret = ZcanCAN_SetAcceptFilter(&paraACFilter);
```

※「IsExtended＝CAN_EID_DISABLE」と設定し ExtendedID を使用しない場合 ExtendedID には「0」が入力されます。（ID,Mask 両方）

※設定は送受信が行われていない状態で設定してください。(コンフィグレーションモードで行うことを推奨します。)

※リセットをかけた場合、全てのアクセプタンスフィルタは無効状態になりますが、ID,Mask の各データは保存されたままになります。

7. RS422 について

本ボードの RS422 はスタート・ストップビットが 1bit、パリティなし、制御なしに固定された機能限定のプロトコルになっています。設定項目が少なくなっているので簡単な通信などにご利用できます。

7.1. 送受信バッファについて

受信には 128 バイトの FIFO バッファが用意されています。また、FIFO バッファには設定した量のデータ数がたまるとフラグを立てたり、割込みをかけることが出来るプログラマブル FIFO（以下、PFIFO）という機能がついています。PFIFO を設定しても 128 バイトの FIFO サイズ自体は変わらないので、PFIFO がフルでも FIFO のサイズまでデータは保存されていきます。

受信したデータは設定した数だけ FIFO バッファから取り出すことになります。指定した数に足りない場合はエラーを返しますが、途中まで FIFO から取り出せたデータとその数は取得することが出来ます。

送信には FIFO などのバッファが用意されていません。そのためデバイスドライバ内で1バイト送信するたびにデータを渡すという処理になりますので、指定した数のデータを送り終わるまでユーザプログラムに戻りません。ボーレートが遅く大量のデータを送信する場合は時間がかかることになるので注意してください。

7.2. ボーレートについて

ボーレートは 13.824MHz を基準として下記式で算出し設定するか、すでに計算された値の設定コードをいくつか用意しているのでそれを利用してください。

BaudRate = 13.824MHz×(1/n+1)[bps]　　(0 <= n < 255)

例. RS422 の設定

```
paraRS422Config.no       = 0;
paraRS422Config.ch       = RS422_CH1;
paraRS422Config.BaudRate = RS422_BR_921_6K;
ZcanRS422_SetConfig(&paraRS422Config);
```

8. リファレンス

8.1. 関数

各機能を使用するための関数を用意しておりますのでオリジナルのプログラム作成にご利用ください。関数を利用するには Zc_IO.o が必要ですのでこのモジュールの組込みと Zc_IO.h のインクルードも忘れないで下さい。(詳しくは sample の Makefile を参照してください。)

8.2. 関数一覧

各機能を使用するための関数を説明します。

表2. 全関数リスト

ボードアクセス関数		
関数	**機能**	**ページ**
Zcan_Open	デバイスのオープン	14
Zcan_Close	デバイスのクローズ	15
Zcan_Info	ボード情報の取得	〃
Zcan_Read_ROTARY_CODE_SWITCH	ボード識別ロータリーコードスイッチの値取得	16
CAN アクセス関数		
関数	**機能**	**ページ**
ZcanCAN_Reset	リセット	16
ZcanCAN_Enable	送受信許可を有効	17
ZcanCAN_Disable	送受信許可を無効	〃
ZcanCAN_SetConfig	通信コンフィグの設定	18
ZcanCAN_GetConfig	通信コンフィグの取得	〃
ZcanCAN_GetStatus	通信状態の取得	19
ZcanCAN_SetACFilter	アクセプタンスフィルタの設定	20
ZcanCAN_GetACFilter	アクセプタンスフィルタの取得	21
ZcanCAN_SendMessage	メッセージの送信	22
ZcanCAN_GetMessage	メッセージの受信	23
ZcanCAN_GetIntStatus	割込み状態の取得	〃
ZcanCAN_SetIntEnable	割込み許可の設定	24
ZcanCAN_GetIntEnable	割込み許可の取得	〃
ZcanCAN_SetIntClear	割込みクリアの設定	25
ZcanCAN_GetErrorStatus	エラー状態の取得	〃
ZcanCAN_SetErrorClear	エラークリア	26

表2.の続き

RS422 アクセス関数		
関数	**機能**	ページ
ZcanRS422_Enable	送信許可を有効	26
ZcanRS422_Disable	送信許可を無効	27
ZcanRS422_SetConfig	通信コンフィグの設定	〃
ZcanRS422_GetConfig	通信コンフィグの取得	28
ZcanRS422_GetStatus	通信状態の取得	〃
ZcanRS422_SendData	データの送信	29
ZcanRS422_GetData	データの受信	〃
ZcanRS422_SetIntEnable	割込み許可の設定	30
ZcanRS422_GetIntEnable	割込み許可の取得	〃
ZcanRS422_SetIntClear	割込みクリアの設定	31
ZcanRS422_SetFifoClear	受信 FIFO バッファのクリア	〃

8.3. 各関数の詳細

Zcan_Open

本ボードにアクセスするドライバを開始します。

書式 int Zcan_Open(int dNo)

引数 dNo･･･デバイス番号(0～3まで指定可能)

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番をオープンします。

```
if( Zcan_Open( 0 ) != RTN_ZCNOK ) printf( “open error” );
```

※デバイス番号は複数枚ささっていた場合の仮割り当て番号になります。ドライバへのアクセスはこの番号で行いますが、ボードの識別にはこの番号とこの先で説明するロータリーコードスイッチ番号を併用してください。

Zcan_Close

本ボードにアクセスするドライバを終了します。

書式 int Zcan_Close(int dNo)

引数 dNo･･･デバイス番号(0～3まで指定可能)

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番をクローズします。

```
if( Zcan_Close( 0 ) != RTN_ZCNOK ) printf( “close error” );
```

Zcan_Info

本ボードのボード情報を取得します。

書式 int Zcan_Info(PZCAN_PARAINFO paraInfo)

引数 paraInfo･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番のボード情報を取得します。

```
ZCAN_PARAINFO paraInfo;
if( Zcan_Info( &paraInfo ) == RTN_ZCNOK ) {
  printf("Serial No.%d¥n",parainfo.DevInfo.ID );
}
```

Zcan_Read_ROTARY_CODE_SWITCH

本ボードのロータリーコードスイッチ番号を取得します。

(ボードの識別に使用してください。)

書式 int Zcan_Read_ROTARY_CODE_SWITCH(int dNo, BYTE *rcsw)

引数 dNo･･･デバイス番号(0～3まで指定可能)

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番のロータリーコードスイッチ番号を取得します。

```
BYTE rc;
If( Zcan_Read_ROTARY_CODE_SWITCH( 0, &rc ) == RTN_ZCNOK ) {
    printf("SW= (%02X)", rc );
}
```

ZcanCAN_Reset

指定した CAN チャンネルをリセットします。

FIFO の全てのデータと通信コンフィグをクリアします。リセット後はコンフィグモードになっています。

書式 int ZcanCAN_Reset(PZCAN_PARA_SET paraSet)

引数 paraSet･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 をリセットします。

```
ZCAN_PARA_SET paraSet;
paraSet.no    = 0;
paraSet.ch    = CAN_CH1;
ZcanCAN_Reset(&paraSet);
```

ZcanCAN_Enable

指定した CAN チャンネルの送受信許可を有効にします。

初期状態は送受信許可になっています。

書式 int ZcanCAN_Enable(PZCAN_PARASET paraSet)

引数 paraSet･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 を送受信許可にします。

```
ZCAN_PARASET paraSet;
paraSet.no = 0;
paraSet.ch = CAN_CH1;
ZcanCAN_Enable(&paraSet);
```

ZcanCAN_Disable

指定した CAN チャンネルの送受信許可を無効にします。

書式 int ZcanCAN_Disable(PZCAN_PARASET paraSet)

引数 paraSet･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 を送受信禁止にします。

```
ZCAN_PARASET paraSet;
paraSet.no = 0;
paraSet.ch = CAN_CH1;
ZcanCAN_Disable(&paraSet);
```

ZcanCAN_SetConfig

指定した CAN チャンネルに通信コンフィグを設定します。

書式 int ZcanCAN_SetConfig(PZCAN_PARA_CONFIG paraConfig)

引数 paraConfig･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 に通信コンフィグを設定します。

```
ZCAN_PARA_CONFIG paraConfig;
paraConfig.no       = 0;
paraConfig.ch       = CAN_CH1;
paraConfig.Mode     = MODE_CAN_LBACK; // ループバックモード
paraConfig.BaudRate = CAN_BR_1M;      // CAN ボーレート 1Mbps
ZcanCAN_SetConfig(&paraConfig);
```

ZcanCAN_GetConfig

指定した CAN チャンネルから通信コンフィグを取得します。

書式 int ZcanCAN_GetConfig(PZCAN_PARA_CONFIG paraConfig)

引数 paraConfig･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 から通信コンフィグを取得します。

```
ZCAN_PARA_CONFIG paraConfig;
paraConfig.no = 0;
paraConfig.ch = CAN_CH1;
if( ZcanCAN_GetConfig(&paraConfig) == RTN_ZCOK ) {
    printf("Mode Code=0x%04X¥n", paraConfig.Mode );
}
```

ZcanCAN_GetStatus

指定した CAN チャンネルの通信状態を取得します。

書式 int ZcanCAN_GetStatus(PZCAN_PARA_STATUS paraStatus)

引数 paraStatus･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 から通信状態を取得します。

```
ZCAN_PARA_STATUS paraStatus;
paraStatus.no = 0;
paraStatus.ch = CAN_CH1;
if( ZcanCAN_GetStatus(&paraStatus) == RTN_ZCOK ) {
  if( paraStatus.BusState == CAN_ST_BUS_BUSY )
      printf( “CAN Bus busy!” );
  if( paraStatus.ErrorState == CAN_ST_ERR_BUSOFF )
      printf( “CAN Bus off state!” );
}
```

ZcanCAN_SetACFilter

指定した CAN チャンネルのアクセプタンスフィルタを設定します。

書式 int ZcanCAN_SetACFilter(PZCAN_PARA_ACFILTER paraACFilter)

引数 paraACFilter･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 にアクセプタンスフィルタを設定します。

```
ZCAN_PARA_ACFILTER paraACFilter;
paraACFilter.no              = 0;
paraACFilter.ch              = CAN_CH1;
paraACFilter.FilterNo        = CAN_ACF_NO1;
paraACFilter.IsEnable        = CAN_ACF_ENABLE;

paraACFilter.Mask.StandardID = 0x03F;
paraACFilter.Mask.ExtendedID = 0x000FF;
paraACFilter.Mask.IsExtended = CAN_MASK_ENABLE;
paraACFilter.Mask.FrameType  = CAN_MASK_DISABLE;

paraACFilter.ID.StandardID   = 0x595;
paraACFilter.ID.ExtendedID   = 0x25555;
paraACFilter.ID.IsExtended   = CAN_EID_ENABLE;
paraACFilter.ID.FrameType    = CAN_FRAME_DATA;
ZcanCAN_SetACFilter(&paraACFilter);
```

ZcanCAN_GetACFilter

指定した CAN チャンネルのアクセプタンスフィルタを取得します。

書式 int ZcanCAN_GetACFilter(PZCAN_PARA_ACFILTER paraACFilter)

引数 paraACFilter･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 からアクセプタンスフィルタ番号1の設定を取得します。

```
ZCAN_PARA_ACFILTER paraACFilter;
paraACFilter.no = 0;
paraACFilter.ch = CAN_CH1;
paraACFilter.FilterNo = CAN_ACF_NO1;

ZcanCAN_GetACFilter(&paraACFilter);
```

ZcanCAN_SendMessage

指定した CAN チャンネルにメッセージを送信させます。

書式 int ZcanCAN_SendMessage(PZCAN_PARA_MESSAGE paraMessage)

引数 paraMessage･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 から拡張 ID を有効にしたデータフレーム送信します。

```
ZCAN_PARA_MESSAGE paraMessage;
paraMessage.no          = 0;
paraMessage.ch          = CAN_CH1;
paraMessage.TxBufType   = CAN_TX_FIFO;     // TX FIFO バッファ
paraMessage.IsExtended  = CAN_ID_IDEON;    // 拡張 ID を有効
paraMessage.FrameType   = CAN_FRAME_DATA;  // データフレーム
paraMessage.StandardID  = 0x259;           // 標準 ID
paraMessage.ExtendedID  = 0x1234;          // 拡張 ID
paraMessage.DataLength  = 8;               // データ長
paraMessage.Data[0]     = 0x12;            // 送信データ
paraMessage.Data[1]     = 0x34;
paraMessage.Data[2]     = 0x56;
paraMessage.Data[3]     = 0x78;
paraMessage.Data[4]     = 0x9a;
paraMessage.Data[5]     = 0xbc;
paraMessage.Data[6]     = 0xde;
paraMessage.Data[7]     = 0xf0;
ZcanCAN_SendMessage(&paraMessage);
```

ZcanCAN_GetMessage

指定した CAN チャンネルから受信したメッセージを取得します。

書式 int Zcan_GetMessage(PZCAN_PARA_MESSAGE paraMessage)

引数 paraMessage･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 から受信した ID やデータを取得します。

```
ZCAN_PARA_MESSAGE paraMessage;
paraMessage.no  = 0;
paraMessage.ch  = CAN_CH1;
if(  ZcanCAN_GetMessage(&paraMessage) == RTN_ZCOK ) {
    printf( “Standard ID=%02X”, paraMessage.StandardID );
}
```

ZcanCAN_GetIntStatus

指定した CAN チャンネルから割込み状態を取得します。

割り込み以外にメッセージの受信チェックにも使用できます。

書式 int ZcanCAN_GetIntStatus(PZCAN_PARA_INTERRUPT paraInterrupt)

引数 paraInterrupt･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 から割り込み状態を取得します。

```
ZCAN_PARA_CONFIG paraInterrupt;
paraInterrupt.no   = 0;
paraInterrupt.ch   = CAN_CH1;
if( ZcanCAN_GetIntStatus(&paraInterrupt) == RTN_ZC_OK ) {
    if( paraInterrupt.Condition & CAN_INT_RXOK )
        printf( “CAN Receive New Message.” );
}
```

ZcanCAN_SetIntEnable

指定した CAN チャンネルに割込み許可を設定します。

書式 int ZcanCAN_SetIntEnable(PZCAN_PARA_INTERRUPT paraInterrupt)

引数 paraInterrupt･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 に新しいメッセージの受信と送信成功の割り込みが起きるよう設定します。

```
ZCAN_PARA_CONFIG paraInterrupt;
paraInterrupt.no    = 0;
paraInterrupt.ch    = CAN_CH1;
paraInterrupt.Condition = CAN_INT_RXOK | CAN_INT_TXOK;
ZcanCAN_SetIntEnable(&paraInterrupt);
```

ZcanCAN_GetIntEnable

指定したチャンネルの割込み許可を取得します。

書式 int ZcanCAN_GetIntEnable(PZCAN_PARA_INTERRUPT paraInterrupt)

引数 paraInterrupt･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 から割込み許可を取得します。

```
ZCAN_PARA_CONFIG paraInterrupt;
paraInterrupt.no    = 0;
paraInterrupt.ch    = CAN_CH1;
if( ZcanCAN_GetIntEnable(&paraInterrupt) == RTN_ZCOK ) {
    printf( “CAN Interrupt Enable Condition = %04X”,
            paraInterrupt.Condition );
}
```

ZcanCAN_SetIntClear

指定した CAN チャンネルの割込み状態をクリアします。

書式 int ZcanCAN_SetIntClear(PZCAN_PARA_INTERRUPT paraInterrupt)

引数 paraInterrupt･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 から新しいメッセージの受信と送信成功の割り込み状態をクリアします。

```
ZCAN_PARA_CONFIG paraInterrupt;
paraInterrupt.no    = 0;
paraInterrupt.ch    = CAN_CH1;
paraInterrupt.Condition = CAN_INT_RXOK | CAN_INT_TXOK;
ZcanCAN_SetIntClear(&paraInterrupt);
```

ZcanCAN_GetErrorStatus

指定した CAN チャンネルのエラー状態を取得します。

書式 int ZcanCAN_GetErrorStatus(PZCAN_PARA_ERROR paraError)

引数 paraError･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 からエラー状態を取得します。

```
ZCAN_PARA_ERROR paraError;
paraError.no    = 0;
paraError.ch    = CAN_CH1;
if( ZcanCAN_GetErrorStatus(&paraError) == RTN_ZCOK ) {
    if( paraError.Status & CAN_ERR_CRC )
        printf( “CAN CRC Error!” );
}
```

ZcanCAN_SetErrorClear

指定した CAN チャンネルのエラー状態をクリアします。

書式 int ZcanCAN_SetErrorClear(PZCAN_PARA_SET paraSet)

引数 paraSet･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の CAN Ch1 にエラー状態をクリアします。

```
PZCAN_PARA_SET paraSet;
paraSet.no    = 0;
paraSet.ch    = CAN_CH1;
ZcanCAN_SetErrorClear(&paraSet);
```

ZcanRS422_Enable

指定した RS422 チャンネルの送信許可を有効にします。

初期状態は送信許可になっています。

書式 int ZcanRS422_Enable(PZCAN_PARASET paraSet)

引数 paraSet･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 を送信許可にします。

```
ZCAN_PARASET paraSet;
paraSet.no = 0;
paraSet.ch = RS422_CH1;
ZcanRS422_Enable(&paraSet);
```

ZcanRS422_Disable

指定した RS422 チャンネルの送信許可を無効にします。

書式 int ZcanRS422_Disable(PZCAN_PARASET paraSet)

引数 paraSet･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 を送信禁止にします。

```
ZCAN_PARASET paraSet;
paraSet.no = 0;
paraSet.ch = RS422_CH1;
ZcanRS422_Disable(&paraSet);
```

ZcanRS422_SetConfig

指定した RS422 チャンネルの通信コンフィグを設定します。

書式 int ZcanRS422_SetConfig(PZCAN_PARA_RS422CONFIG paraConfig)

引数 paraConfig･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 に通信コンフィグを設定します。

```
ZCAN_PARA_RS422CONFIG paraConfig;
paraConfig.no = 0;
paraConfig.ch = RS422_CH1;
paraConfig.RxPFifoByte = 4;          //プログラマブル FIFO 4byte
paraConfig.BaudRate = RS422_BR_921_6K; //ボーレート 921.6kbps
ZcanRS422_SetConfig(&paraConfig);
```

ZcanRS422_GetConfig

指定した RS422 チャンネルから通信コンフィグを取得します。

書式 int ZcanRS422_GetConfig(PZCAN_PARA_RS422CONFIG paraConfig)

引数 paraConfig･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 から通信コンフィグを取得します。

```
ZCAN_PARA_RS422CONFIG paraConfig;
paraConfig.no = 0;
paraConfig.ch = RS422_CH1;
if( ZcanRS422_GetConfig(&paraConfig) == RTN_ZCOK ) {
    printf( “RS422 Receive Programable FIFO Size = %d byte”,
              paraConfig.RxPFifoByte );
}
```

ZcanRS422_GetStatus

指定した RS422 チャンネルの通信状態を取得します。

書式 int ZcanRS422_GetStatus(PZCAN_PARA_RS422FLAG paraFlag)

引数 paraFlag･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 から通信状態を取得します。

```
ZCAN_PARA_RS422FLAG paraFlag;
paraFlag.no = 0;
paraFlag.ch = RS422_CH1;
if( ZcanRS422_GetStatus(&paraFlag) == RTN_ZCOK ) {
    if( paraFlag.RxError ) printf( “RS422 Receive error!!” );
}
```

ZcanRS422_SendData

指定した RS422 チャンネルにデータを送信させます。

書式 int ZcanRS422_SendData(PZCAN_PARA_RS422DATA paraData)

引数 paraData･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 に指定したデータを送信させます。

```
ZCAN_PARA_RS422DATA paraData;
paraData.no = 0;
paraData.ch = RS422_CH1;
paraData.byte = 2;              //1～128 バイトまで
paraData.Buffer[0] = 0x55;
paraData.Buffer[1] = 0xAA;
ZcanRS422_SendData(&paraData);
```

ZcanRS422_GetData

指定した RS422 チャンネルから受信したデータを設定したバイト分取得します。

データが足りない場合はエラーを返しますが取得できた分のデータは確保され、そのデータ数も取得できます。

書式 int ZcanRS422_GetData(PZCAN_PARA_RS422DATA paraData)

引数 paraData･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 から受信したデータを取得します。

```
ZCAN_PARA_RS422DATA paraData;
paraData.no = 0;
paraData.ch = RS422_CH1;
paraData.byte = 3;
if( ZcanRS422_GetData(&paraData) == RTN_ZCOK ) {
    printf( "Get data[0] = %02X¥n", paraData.Buffer[0] );
}
```

ZcanRS422_SetIntEnable

指定した RS422 チャンネルの割込み許可を設定します。

設定できる割込みは PFIFO フルと受信エラーのみとなります。

書式 int ZcanRS422_SetIntEnable(PZCAN_PARA_RS422FLAG paraFlag)

引数 paraFlag･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 に PFIFO フル割込みと受信エラー割込みを許可します。

```
ZCAN_PARA_RS422FLAG paraFlag;
paraFlag.no        = 0;
paraFlag.ch        = RS422_CH1;
paraFlag.RxPFFull  = RS422_INT_ENABLE;
paraFlag.RxError   = RS422_INT_ENABLE;
ZcanRS422_SetIntEnable(&paraFlag);
```

ZcanRS422_GetIntEnable

指定した RS422 チャンネルの割込み許可を取得します。

取得できる割込みは PFIFO フルと受信エラーのみとなります。

書式 int ZcanRS422_GetIntEnable(PZCAN_PARA_RS422FLAG paraFlag)

引数 paraFlag･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 Ch1 から割り込み許可を取得します。

```
ZCAN_PARA_RS422FLAG paraFlag;
paraFlag.no = 0;
paraFlag.ch = RS422_CH1;
if( ZcanRS422_GetIntEnable(&paraFlag) == RTN_ZCOK ) {
    if( paraRS422Flag.RxPFFull )
        printf( “RS422 Receive PFIFO Full Interrupt Enable.” );
}
```

ZcanRS422_SetIntClear

指定した RS422 チャンネルの割込み状態をクリアします。

クリアできる割込みは PFIFO フルと受信エラーのみとなります。

書式 int ZcanRS422_SetIntClear(PZCAN_PARA_RS422FLAG paraFlag)

引数 paraFlag･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 1ch の PFIFO バッファのフル割込みとエラー割込みをクリアします。

```
ZCAN_PARA_RS422FLAG paraFlag;
paraFlag.no       = 0;
paraFlag.ch       = RS422_CH1;
paraFlag.RxPFFull = RS422_CLR_ENABLE;  //RX PFIFO Full clear
paraFlag.RxError  = RS422_CLR_ENABLE;  //RX Error clear
ZcanRS422_SetIntClear(&paraFlag);
```

ZcanRS422_SetFifoClear

指定した RS422 チャンネルの受信 FIFO バッファをクリアします。

書式 int ZcanRS422_SetFifoClear(PZCAN_PARA_SET paraSet)

引数 paraSet･･･データ形式参照

戻り値 成功:RTN_ZCNOK, 0
失敗:0 以外

＜例＞ デバイス0番の RS422 1ch から受信 FIFO バッファをクリアします。

```
ZCAN_PARA_SET paraSet;
paraSet.no = 0;
paraSet.ch = RS422_CH1;
ret = ZcanRS422_SetFifoClear(&paraSet);
```

8.4. データ形式

8.2.**各関数の詳細**を利用する為には機能別に使用するためのデータ形式が必要です。
データ形式の基本は構造体で、型名としてユーザで宣言し使用することが出来ます。

8.5. データ形式一覧

本ボードで使用されるデータ形式の一覧です。

表3. 全データ形式リスト

ボードアクセス用データ形式			
型名	ポインタ型名	説明	ページ
ZCAN_DEV_INFO	PZCAN_DEV_INFO	デバイス情報	33
ZCAN_DEV	PZCAN_DEV	ボード情報	〃
ZCAN_PARA_SET	PZCAN_PARA_SET	CAN/RS422 チャンネル設定	〃
CAN アクセス用データ形式			
型名	ポインタ型名	説明	ページ
ZCAN_PARA_CONFIG	PZCAN_PARA_CONFIG	通信コンフィグ	34
ZCAN_PARA_STATUS	PZCAN_PARA_STATUS	通信状態	〃
ZCAN_PARA_ID	PZCAN_PARA_ID	アクセプタンスフィルタ用 ID	35
ZCAN_PARA_ACFILTER	PZCAN_PARA_ACFILTER	アクセプタンスフィルタ設定	〃
ZCAN_PARA_MESSAGE	PZCAN_PARA_MESSAGE	送受信メッセージ	36
ZCAN_PARA_ERROR	PZCAN_PARA_ERROR	エラー状態	〃
ZCAN_PARA_INTERRUPT	PZCAN_PARA_INTERRUPT	割込み状態	〃
RS422 アクセス用データ形式			
型名	ポインタ型名	説明	ページ
ZCAN_PARA_RS422CONFIG	PZCAN_PARA_RS422CONFIG	通信コンフィグ	37
ZCAN_PARA_RS422FLAG	PZCAN_PARA_RS422FLAG	通信状態フラグ	〃
ZCAN_PARA_RS422DATA	PZCAN_PARA_RS422DATA	送受信データ	〃

8.6. 各データ形式の詳細

この章では各データ形式の型名やその中のメンバ変数について説明いたします

※備考で説明しているコードは **8.6.各種設定／戻り値コード**で詳細を確認してください。

ZCAN_DEV_INFO, PZCAN_DEV_INFO

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	irq	IRQ 番号
	WORD	Ver	PCI バージョン
	WORD	ID	シリアル No.の下 4 桁
	DWORD	ioBase	ベースアドレス 2
	DWORD	ioLCR	ベースアドレス 0
備考	接続されている本ボードについての情報です。ID はシリアル No の下4桁の数字で、そこからボードの判別を行うことが出来ます。		

ZCAN_DEV, PZCAN_DEV

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	ZCAN_DEV_INFO	DevInfo	デバイス情報
備考			

ZCAN_PARA_SET, PZCAN_PARA_SET

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	CAN/RS422 チャンネル番号(0～9)
備考	チャンネルのみを指定する関数用		

ZCAN_PARA_CONFIG, PZCAN_PARA_CONFIG

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	CAN チャンネル番号(0～9)
	BYTE	Mode	モード
	BYTE	BaudRate	ボーレートコード
	WORD	Prescaler	ボーレートプリスケーラ(0～255)
	BYTE	SyncJumpWidth	同期ずれ補償の最大幅(0～3)
	BYTE	TimeSegment1	プロパゲーションとフェーズ1の合計(0～15)
	BYTE	TimeSegment2	フェーズセグメント2(0～7)

備考 モードは CAN_MODE_・・・、ボーレートコードは CAN_BR_・・・のコードから選択してください。
ボーレートコードを使用しない場合は「CAN_BR_NULL」を必ず入力してください。

ZCAN_PARA_STATUS, PZCAN_PARA_STATUS

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	CAN チャンネル番号(0～9)
	BYTE	Mode	現在のモード
	BYTE	BusState	バス状態
	BYTE	ErrorState	エラー状態
	BYTE	ErrorWarning	エラー警告
	BYTE	TxHpbFull	最優先送信バッファのフルフラグ
	BYTE	TxFifoFull	FIFO 送信バッファがフルフラグ
	BYTE	ACFBusy	アクセプタンスフィルタの動作状態

備考 モードは CAN_MODE_・・・、バス状態は CAN_ST_BUS_・・・、エラー状態は CAN_ST_ERR_・・・のコードから選択してください。
エラー警告は送受信エラーカウンタがどちらかが 96 以上を超えた時「1」になります。それ以外は「0」です。
残りの3つのメンバはその状態になると「1」になります。それ以外は「0」です。

ZCAN_PARA_ID, PZCAN_PARA_ID

メンバリスト	型名	メンバ名	説明
	WORD	StandardID	標準 ID(0～2047)
	DWORD	ExtendedID	拡張 ID(0～262143)
	BYTE	IsExtended	拡張 ID の有効／無効
	BYTE	FrameType	送受信フレームタイプ

備考 アクセプタンスフィルタのためのデータ形式です。他で使用することはありません。

ZCAN_PARA_ACFILTER, PZCAN_PARA_ACFILTER

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	CAN チャンネル番号(0～9)
	WORD	FilterNo	フィルタ番号(0～3)
	BYTE	IsEnable	フィルタの有効／無効
	ZCAN_PARA_ID	Mask	マスク ID
	ZCAN_PARA_ID	ID	比較用 ID

備考 フィルタの有効／無効は CAN_ACF_・・・のコードから選択してください。
フィルタ番号は CAN_ACF_NOn・・・のコードからも選択可能です。
マスク ID 内の拡張 ID の有効／無効と送受信フレームタイプは CAN_MASK_・・・のコードから選択してください。
比較用 ID 内の拡張 ID の有効／無効は CAN_EID_・・・、送受信フレームタイプは CAN_FRAME_・・・のコードから選択してください。

ZCAN_PARA_MESSAGE, PZCAN_PARA_MESSAGE

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	CAN チャンネル番号(0～9)
	WORD	StandardID	標準 ID(0～2047)
	DWORD	ExtendedID	拡張 ID(0～262143)
	BYTE	TxBufType	送信バッファタイプ
	BYTE	IsExtended	拡張 ID の有効／無効
	BYTE	FrameType	送受信フレームタイプ
	BYTE	DataLength	送受信データ長(0～8)
	BYTE	Data[CAN_DATA_SIZE]	送受信データ

備考 送受信バッファタイプは CAN_TX_・・・、拡張 ID の有効／無効は CAN_EID_・・・、送受信フレームタイプは CAN_FRAME_・・・のコードから選択してください。

ZCAN_PARA_ERROR, PZCAN_PARA_ERROR

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	CAN チャンネル番号(0～9)
	BYTE	Status	エラー状態レジスタ(5bit)
	BYTE	TxCounter	送信エラーカウンタ
	BYTE	RxCounter	受信エラーカウンタ

備考 エラー状態レジスタは下位 5bit の各ビットの状態でエラー内容がわかるようになっています。CAN_ERR_・・・のコードから確認したい状態を選択して「&」をとり、「0」でなければそのエラー状態になっています。

ZCAN_PARA_INTERRUPT, PZCAN_PARA_INTERRUPT

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	CAN チャンネル番号(0～9)
	WORD	Condition	割り込み条件レジスタ(11bit)

備考 割り込み条件レジスタは下位 11bit の各ビットの状態で割り込みの条件がわかります。CAN_INT_・・・のコードから確認・設定したい条件を選択して「&」をとり、「0」でなければその条件が割り込み状態・設定されていることになります。

ZCAN_PARA_RS422CONFIG, PZCAN_PARA_RS422CONFIG

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	RS422 チャンネル番号(0～1)
	BYTE	RxPFifoByte	PFIFO(1～127、初期値 12)
	WORD	BaudRate	ボーレート

備考 ボーレートは RS422 ボーレート計算値を入力するか RS422_BR_・・・のコードから選択してください。

ZCAN_PARA_RS422FLAG, PZCAN_PARA_RS422FLAG

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	RS422 チャンネル番号(0～1)
	WORD	RxFifoFull	FIFO 受信バッファフルフラグ
	DWORD	RxPFFull	PFIFO 受信バッファフルフラグ
	BYTE	RxError	受信エラーフラグ
	BYTE	RxEmpty	FIFO 受信バッファ空フラグ

備考 割込みや通信状態のフラグに使用します。関数によっては使用しないメンバもあります。(詳細は関数を参照。)
全てのフラグは「1」でその状態となります。それ以外は「0」となります。

ZCAN_PARA_RS422DATA, PZCAN_PARA_RS422DATA

メンバリスト	型名	メンバ名	説明
	WORD	no	デバイス番号(0～3)
	WORD	ch	RS422 チャンネル番号(0～1)
	BYTE	byte	送受信データ数(1～128)
	BYTE	Buffer[RS422_BUFFER_SIZE]	送受信データ

備考

8.7. 各種設定／戻り値コード

各データ形式のメンバなどで利用できる設定コード一覧です。

表4. CAN 関連設定コード

コード名	値	説明
CAN_MODE_CONFIG	0x08	コンフィグレーション
CAN_MODE_LBACK	0x04	ループバック
CAN_MODE_SLEEP	0x02	スリープ
CAN_MODE_NORMAL	0x01	通常動作
CAN_BR_NULL	0	ボーレートコードの使用不可
CAN_BR_1M	1	1Mbps
CAN_BR_500K	2	500kbps
CAN_BR_250K	3	250kbps
CAN_BR_125K	4	125kbps
CAN_BR_83_3K	5	83.3kbps
CAN_BR_33_3K	6	33.3kbps
CAN_TX_FIFO	0	FIFO 送信バッファ
CAN_TX_HPB	1	最優先送信バッファ
CAN_EID_DISABLE	0	拡張 ID が無効
CAN_EID_ENABLE	1	拡張 ID が有効
CAN_FRAME_DATA	0	送受信データフレーム
CAN_FRAME_REMOTE	1	送受信リモートフレーム
CAN_DATA_SIZE	8	CAN データの最大数
CAN_CH1	0	CAN チャンネル1
CAN_CH2	1	〃2
CAN_CH3	2	〃3
CAN_CH4	3	〃4
CAN_CH5	4	〃5
CAN_CH6	5	〃6
CAN_CH7	6	〃7
CAN_CH8	7	〃8
CAN_CH9	8	〃9
CAN_CH10	9	〃10
CAN_CHMAX	10	〃数
CAN_ACF_NO1	0	アクセプタンスフィルタ番号 1
CAN_ACF_NO2	1	〃2
CAN_ACF_NO3	2	〃3
CAN_ACF_NO4	3	〃4
CAN_ACF_NOMAX	4	〃数
CAN_ACF_DISABLE	0	アクセプタンスフィルタ無効
CAN_ACF_ENABLE	1	アクセプタンスフィルタ有効

表4. の続き

コード名	値	説明
CAN_MASK_DISABLE	0	マスク無効
CAN_MASK_ENABLE	1	マスク有効
CAN_ST_BUS_CONFIG	0x00	コンフィグレーション中
CAN_ST_BUS_BUSY	0x01	バス動作中
CAN_ST_BUS_IDLE	0x02	アイドル中
CAN_ST_BUS_ERROR	0x03	エラー
CAN_ST_ERR_NONE	0x00	エラー状態なし
CAN_ST_ERR_ACTIVE	0x01	〃 アクティブ
CAN_ST_ERR_BUSOFF	0x02	〃 バスオフ
CAN_ST_ERR_PASSIVE	0x03	〃 パッシブ
CAN_ERR_ACK	0x01	ACK エラー
CAN_ERR_BIT	0x02	ビットエラー
CAN_ERR_STUFF	0x04	スタッフエラー
CAN_ERR_FORM	0x08	フォームエラー
CAN_ERR_CRC	0x10	CRC エラー
CAN_INT_WAKEUP	0x0001	CAN 割込み条件／ウェークアップ
CAN_INT_SLEEP	0x0002	〃 スリープ
CAN_INT_BUSOFF	0x0004	〃 バスオフ
CAN_INT_ERROR	0x0008	〃 エラー
CAN_INT_RXNEMP	0x0010	〃 受信 FIFO が空
CAN_INT_RXOFLW	0x0020	〃 受信 FIFO のオーバーフロー
CAN_INT_RXUFLW	0x0040	〃 受信 FIFO のアンダーフロー
CAN_INT_RXOK	0x0080	〃 新しいメッセージを受信
CAN_INT_TXBFLL	0x0100	〃 最優先送信バッファが満杯
CAN_INT_TXFLL	0x0200	〃 送信 FIFO が満杯
CAN_INT_TXOK	0x0400	〃 送信成功
CAN_INT_ARBLST	0x0800	〃 送信時のバスアービトレーション負け

表5. RS422 関連設定コード

コード名	値	内容
RS422_CH1	0	RS422 チャンネル1
RS422_CH2	1	〃 2
RS422_CHMAX	2	〃 数
RS422_BUFFER_SIZE	128	バッファサイズ
RS422_INT_ENABLE	1	割り込み有効
RS422_INT_DISABLE	0	割り込み無効
RS422_CLR_ENABLE	1	クリア有効
RS422_CLR_DISABLE	0	クリア無効

表5. の続き

コード名	値	内容
RS422_STT_RXEMP	0x0001	FIFO 受信バッファ空フラグ
RS422_STT_RXPFLL	0x0002	PFIFO 受信バッファフルフラグ
RS422_STT_RXFLL	0x0004	FIFO 受信バッファフルフラグ
RS422_STT_RXERR	0x0008	受信エラーフラグ
RS422_STT_TXBUSY	0x0100	送信中フラグ
RS422_BR_1_536M	8	1.536Mbps
RS422_BR_921_6K	14	921.6kbps
RS422_BR_576K	23	576kbps
RS422_BR_384K	35	384kbps
RS422_BR_288K	47	288kbps
RS422_BR_192K	71	192kbps
RS422_BR_144K	95	144bps
RS422_BR_115_2K	119	115.2kbps
RS422_BR_96K	143	96kbps
RS422_BR_57_6K	239	57.6kbps

各関数の戻り値コード一覧です。

表6. 戻り値コード

コード名	値	内容
RTN_ZCOK	0	正常
RTN_ZCOVER	1	本ボードが無い(0～3以外)
RTN_ZCNOPEN	2	本ボードがオープンされていない
RTN_ZCNOCLOSE	3	本ボードがクローズされない
RTN_ZCNOPARA	4	パラメータが指定されていない
RTN_ZCCHOVER	5	チャンネルが間違っている
RTN_ZCTIMEOUT	6	タイムアウト
RTN_ZCPARAERROR	7	パラメータにエラー
RTN_ZCRXERROR	8	受信エラー
RTN_ZCRXEMPTY	9	受信バッファエラー
RTN_ZCRXOFLW	10	受信データオーバーフロー
RTN_ZCTXOFLW	11	送信データオーバーフロー
RTN_ZCBUFOFLW	12	バッファオーバーフロー

9. サポート

本ソフトウェアのお問合せは

有限会社　図工

TEL/0463-97-4891，FAX/0463-97-4895，e-mail/support@zuco.jp

質問などは下記点について明記していただきますと助かります。

1) お使いのシステム(PCスペックなど)

2) OSのディストリビューションとカーネルバージョン

3) エラー出力された場合はその内容

バージョンアップの情報や修正などの内容についてはホームページ上で最新のものを更新しております。

10. 改訂履歴

版	日付	ページ	内容
1	2008/6/13	ー	初版作成
2	2008/6/27	-	関数・データ形式の追加・修正
3	2008/7/14	9 34	RS422 のボーレート基準クロックを変更 RS422 のボーレート設定コードを変更
4	2008/8/1	6-7 - -	モードの説明追加 アクセプタンスフィルタの説明・関数・データ形式・設定コードの追加 各章の追加によるページ番号の修正

開発元

独立行政法人　産業技術総合研究所

製造・販売元／修理、技術的なお問い合わせは

有限会社 図工

ＴＥＬ：０４６３－９７－４８９１ ／ ＦＡＸ：０４６３－９７－４８９５

ＵＲＬ：http://www.zuco.jp ／ E-mail：support@zuco.jp