

# ***Vibration Function Library***

""( 仮称)

***Version 0.10***

***Reference and Tutorial***

***1998/10/8 Rev.0.10***

## 目次

### 第1章 振動ライブラリ概要

1.1 特徴 .....	3
1.2 仕様 .....	3
1.3 本ドキュメントの読み方 .....	3

### 第2章 ぶるぶるバック(仮)を振動させる

2.1 ぶるぶるバック(仮)のスペック .....	4
2.2 接続確認 .....	4
2.3 振動させる .....	5
2.4 振動に必要なパラメータと設定例 .....	5
2.5 振動を止める .....	6
2.6 自動停止時間の設定 .....	6
2.7 ぶるぶるバック(仮)の振動に関する関数の説明 .....	7

### 第3章 ぶるぶるバック(仮)以外の一般振動デバイスに対応する

3.1 接続確認 .....	10
3.2 情報取得 .....	10
3.2.1 振動デバイスの接続方向 .....	10
3.2.2 振動ユニット .....	11
3.2.3 振動ユニット数と同時設定可能なユニット数 .....	11
3.2.4 振動ユニット情報の取得 .....	12
3.2.5 振動ユニット情報の詳細 .....	13
3.3 情報取得に関する関数の説明 .....	13
3.4 振動させる .....	16
3.5 振動に必要なパラメータと設定例 .....	17
3.6 振動を止める .....	18
3.7 自動停止時間の設定 .....	18
3.8 振動に関する関数の説明 .....	19

### 第4章 任意振動波形

# 第1章

## 振動ライブラリ概要

### 1.1 特徴

本ライブラリは、コントロールパッド等に接続される「ぶるぶるパック(仮)」をはじめとする Maple バイブレーションファンクションを備えた振動デバイスへのアクセスを可能にする忍拡張ライブラリです。

### 1.2 仕様

本ライブラリの仕様は以下のとおりです。また、この仕様は将来変更の可能性があります。

項目	内容
対応ハードウェア	Maple Bus 仕様バイブレーションファンクションデバイス
対応デバイス数	8

表 1-1.仕様

### 1.3 本ドキュメントの読み方

#### 1.「ぶるぶるパック(仮)」のみに対応するアプリケーションの場合

「ぶるぶるパック(仮)」とは、最初に開発、登場予定の振動デバイスの名称です。これのみに対応すればよい

アプリケーションを作成する場合は、第2章に進んでください。

#### 2. 今後登場する振動デバイスに、なるべくすべて対応したい場合

この場合は、振動させる前に、振動デバイスの情報取得等の下準備が必要となります。

第3章 第4章に進んでください。

いずれの場合も、アプリケーションで取り扱うデバイスは熟知しておく必要がありますので、デバイスの仕様書もあわせて参照するようにしてください。

「ぶるぶるパック(仮)」の名称は、今後変更される場合がありますが、その場合も本ドキュメントでは「ぶるぶるパック(仮)」と表記します。また、「ぶるぶるパック(仮)」の仕様も、同様に変更される場合があります。

## 第2章

# ぶるぶるパック(仮)を振動させる

本ライブラリ関数を適切に使用することにより、コントローラに接続されるさまざまな振動デバイスに対応することが可能ですが、ここでは簡単に「ぶるぶるパック(仮)」（最初に登場予定の振動デバイス）を振動させてみることにしましょう。

### 2.1 ぶるぶるパック(仮)のスペック

ライブラリ関数を利用してアクセスするために必要なスペックのみを記述します。さらに詳細なスペックはぶるぶるパック(仮)ハードウェア仕様書および Maple バイブレーションファンクション仕様書を参照してください。

項目	内容
振動ユニットの個数	1
振動ユニットの位置	前
振動ユニットの振動軸	なし
連続振動	可
振動方向設定(正方向、逆方向)	可
設定可能振動強度	-7 ~ 0 ~ +7
任意振動波形設定	不可
設定可能振動周波数	0FH(15Hz) ~ 4FH(79Hz)(未定)

表 2-1.ぶるぶるパック(仮)仕様

### 2.2 接続確認

ぶるぶるパック(仮)の接続を確認するには、関数 `pdVibMxIsReady()` を使用します。

```
if (pdVibMxIsReady(PDD_PORT_A2)) {  
    /* ぶるぶるパック(仮)が接続されている */  
}
```

## 2.3 振動させる

振動させるには、関数 `pdVibMxStart()` を使用します。この時、振動させるためのパラメータを構造体にセットする必要があります。

```
Sint32 ret;
PDS_VIBPARAM param;

param.unit = 1; /* ユニット番号1(固定) */
param.flag = PDD_VIB_FLAG_CONTINUOUS; /* 連続振動あり */
param.power = 7; /* 正方向に7の強さ(最大)で */
param.freq = 20; /* 20Hz で */
param.inc = 0; /* 収束、発散なし(0Hz) */

ret = pdVibMxStart(PDD_PORT_A2, &param); /* 振動させる */

if (ret != PDD_VIBERR_OK) {
    /* 振動させることができなかった */
}
```

この例では、連続振動あり、最大強度、振動周波数 20Hz で振動させています。初期状態では、連続振動は約 5 秒で自動的に停止します。この自動停止時間は、関数 `pdVibMxSetStopTime()` を使用して変更することができます。

## 2.4 振動に必要なパラメータと設定例

関数 `pdVibMxStart()` に渡す `PDS_VIBPARAM` 構造体の各メンバの意味と、設定例を解説します。

メンバ	意味	指定する値	説明
unit	振動ユニット番号	1	振動パラメータを送る振動ユニットを指定します。 ぶるぶるバック(仮)の場合は、振動ユニットを1つのみ持つので、1 固定で使います。
flag	振動フラグ	PDD_VIB_FLAG_CONTINUOUS	連続振動を指定します。 振動自動停止時間が来るまで振動を続けます。 指定しないと単発振動(ワンショット)となります。
		PDD_VIB_FLAG_EXHALATION	振動がだんだん強くなるよう指定します。(発散振動) PDD_VIB_FLAG_CONVERGENCE とは同時に指定できません。 連続振動時には、最大強度に達すると再び power で指定された強度から発散振動を開始します。
		PDD_VIB_FLAG_CONVERGENCE	振動がだんだん弱くなるよう指定します。(収束振動) PDD_VIB_FLAG_EXHALATION とは同時に指定できません。 連続振動時には、最小強度に達すると再び power で指定された強度から収束振動を開始します。
power	振動の強さ	-7 ~ 0 ~ +7	正で正方向、負で逆方向に振動します。 値が大きいほど強い振動となります。
freq	振動数(振動周波数)	15 ~ 79(予定)	振動数(振動周波数)を指定します。
inc	振動勾配周期	1 ~ 255	収束、発散させる場合の周期を指定します。 収束、発散させない場合は無視されます。

表 2-2.パラメーター一覧

**例 1 . 振動周波数 50Hz で、単発振動**

```
param.unit = 1;
param.flag = 0;
param.power = 7;
param.freq = 50;
param.inc = 0;
```

**例 2 . 振動周波数 20Hz で、収束単発振動**

```
param.unit = 1;
param.flag = PDD_VIB_FLAG_CONVERGENCE;
param.power = -7;
param.freq = 20;
param.inc = 1;
```

**例 3 . 振動周波数 40Hz で、発散連続振動**

```
param.unit = 1;
param.flag = PDD_VIB_FLAG_CONTINUOUS | PDD_VIB_FLAG_EXHALATION;
param.power = 7;
param.freq = 40;
param.inc = 1;
```

振動パラメータに関しては、実際にさまざまな値を設定して実験し、必ず振動を実機で確認してください。

## 2.5 振動を止める

振動を止めるには、関数 `pdVibMxStop()` を使用します。

```
Sint32 ret;
ret = pdVibMxStop(PDD_PORT_A2); /* 振動を止める */

if (ret != PDD_VIBERR_OK) {
    /* 止められなかった */
}
```

次のような方法でも止めることができます。

```
Sint32 ret;
PDS_VIBPARAM param;

param.unit = 1; /* ユニット番号1(固定) */
param.flag = 0; /* フラグなし */
param.power = 0; /* 強さ 0 を指定すると止まる */
param.freq = 0; /* 0Hz で */
param.inc = 0; /* 収束、発散なし */

ret = pdVibMxStart(PDD_PORT_A2, &param); /* 振動を止める */

if (ret != PDD_VIBERR_OK) {
    /* 止められなかった */
}
```

## 2.6 自動停止時間の設定

振動デバイスは、連続振動させる場合に、指定時間で自動停止する機能があります。デバイス

の初期状態では、この時間は約 5 秒となっています。この時間を変更するには関数 `pdVibMxSetStopTime()` を使用します。

```
Sint32 ret;
Uint32 time;

time = 10; /* 自動設定時間を 10(約 2.5 秒)にする */
ret = pdVibMxSetStopTime(PDD_PORT_A2, time); /* 自動停止時間を設定する */

if (ret != PDD_VIBERR_OK) {
    /* 設定できなかった */
}
```

時間パラメータは 0.25 秒刻みで、0x00(約 0.25 秒) ~ 0xff(約 64 秒)まで指定できます。この例では、約 2.5 秒に設定しています。

この自動停止時間は、デバイスへの電源投入時には初期化され、5 秒に設定されますが、それ以外では初期化されません。他のアプリケーションで使用されていた場合などを考慮し、アプリケーションの初期化時に必ず何らかの値を設定するようにしてください。

なお、自動停止を禁止する方法はありません。

## 2.7 ぶるぶるバック(仮)の振動に関する関数の説明

関数	機能
<code>pdVibMxIsReady</code>	ぶるぶるバック(仮)の接続を調べる
<code>pdVibMxStart</code>	振動を開始する
<code>pdVibMxStop</code>	振動を停止する
<code>pdVibMxSetStopTime</code>	振動自動停止時間を設定する

表 2-3.関数一覧

### ・API

#### pdVibMxIsReady

ぶるぶるバック(仮)  
関数

書式	Sint32 pdVibMxIsReady(Uint32 port)
パラメータ	port                      ポート番号                      PDD_PORT_A1/A2/B1/B2/C1/C2/D1/D2
戻り値	TRUE                      ぶるぶるバック(仮)が接続されている FALSE                      ぶるぶるバック(仮)が接続されていない
機能	・ぶるぶるバック(仮)が接続されているかどうかを調べます
参照	
備考	・本バージョンでは、振動デバイスが接続されていれば TRUE を返します。
例	if (pdVibMxIsReady(PDD_PORT_A2)) return TRUE;

#### pdVibMxStart

ぶるぶるバック(仮)  
関数

書式	Sint32 pdVibMxStart(Uint32 port, const PDS_VIBPARAM* param)
パラメータ	port                      ポート番号                      PDD_PORT_A1/A2/B1/B2/C1/C2/D1/D2 param                      振動パラメータ構造体のアドレス
戻り値	PDD_VIBERR_OK                      正常終了 PDD_VIBERR_NO_VIBRATOR                      ぶるぶるバック(仮)は接続されていない PDD_VIBERR_BUSY                      ぶるぶるバック(仮)は BUSY 状態
機能	・ぶるぶるバック(仮)に、振動を開始させます。
参照	PDS_VIBPARAM
備考	・

例

```
param.unit = 1;
param.flag = PDD_VIB_FLAG_CONTINUOUS;

param.freq = 15;
param.inc = 0;

/* */
}
```

				ふるふるパック( ) 関数
パラメータ	Sint32 pdVibMxStop(Uint32 port)	ポート番号	PDD_ _A1/A2/B1/B2/C1/C2/D1/D2	
	PDD_VIBERR_OK		正常終了 ふるふるパック( )は接続されていない ふるふるパック( )は 状態 (仮 に、振動を停止させます。	
機能 参照 備考 例	/* 成功 }			

pdVibMxSetStopTime				(仮 関数
書式				
パラメータ	port	ポート番号	PDD_ _A1/A2/B1/B2/C1/C2/D1/D2 00H ~ 当たり約 0.25 )	
戻り値	PDD_VIBERR_NO_VIBRATOR PDD_VIBERR_BUSY ・ふるふるパック 仮)	正常終了	(仮 は接続されていない (仮 は BUSY	
機能 参照 備考 例	if (pdVibMxSetStopTime(PDD_PORT_A2, 0x10) == PDD_VIBERR_OK) { 成功 */			



## ・構造体

**PDS\_VIBPARAM**構造体

---

定義	typedef struct { Uint8 unit; Uint8 flag; Sint8 power; Uint8 freq; Uint8 inc; Uint8 reserved[3]; } PDS_VIBPARAM;
説明	振動デバイスへの振動パラメータを格納する構造体です
メンバ	unit           振動パラメータを送るユニット番号 flag           振動フラグ power          振動強度 freq          振動周波数 inc           振動勾配周期 reserved       予約
参照	pdVibMxStart()

## 第3章

# ぷるぷるパック(仮 以外の一般振動デバイスに対応する

(仮 に限定した関数の解説を行いました。しかし今後、さまざまな振動デバイスが登場するかも知れません。そのような振動デバイスに対応したい場合、は、振動ユニット  
ます。

### 3.1 接続確認

pdVibIsReady()を使用します。

```
        /* 振動デバイスが接続されている */  
    }
```

### 3.2 情報取得

情報には、振動デバイスの情報(PDS\_VIBINFO)  
(PDS\_VIBUNITINFO)の2種類があります。

#### 3.2.1 振動デバイスの接続方向

コントローラの拡張コネクタの向きによって、ユーザーから見て振動の方向が異なる場合があります。  
たとえば、標準コントローラに振動パックを接続すると、振動の方向は反転してしまいます。

pdVibGetDirection()を使用します。

```
Sint32 dir;  
  
switch (dir) {  
    case PDD_VIB_DIRECTION_          /* 通常  
        break;  
    case PDD_VIB_DIRECTION_          /* 逆さま  
        break;  
    case PDD_VIB_DIRECTION_          /* 左向き  
        break;  
    case PDD_VIB_DIRECTION_          /* 右向き  
        break;  
}
```

### 3.2.2 振動ユニット

振動デバイス内部に存在し、振動源となるハードウェアを、本ライブラリでは「振動ユニット」と呼びます。ユニットは 1 デバイス内に複数存在でき、1 ~ 15 個のユニットがサポートされています。

ユニットには固有のパラメータがあり、次のようなものがあります。

ユニットパラメータ	内容
振動ポジション	ユニットが デバイスのどの位置に設置されているかを示します。前後左右4種類のポジションがあります。
振動軸	ユニットは 振動する軸(向き)を持ちます。軸なし、X 軸、Y 軸、Z 軸の4種類があります。
振動強度	ユニットには 振動の強度が8段階可変のものと、固定のものがあります。
連続振動	ユニットには 次に振動設定をするまで、連続して振動していただける連続振動をサポートしているものがあります。
振動方向	ユニットには、プラスとマイナスの振動方向を指定できるものがあります。
任意振動波形	ユニットには、任意の波形を作成して、そのとおりに振動させることができるものがあります。
振動数(振動周波数)	

表 3-1.ユニットパラメータ

### 3.2.3 振動ユニット数と同時設定可能なユニット数

関数 `pdVibGetInfo()` でユニット数、振動の同時設定が可能なユニット数を取得します。

```
PDS_VIBINFO info;

if (pdVibGetInfo(PDD_PORT_A2, &info)) == PDD_VIBERR_OK) {
    /* 情報取得に成功 */
    printf("振動ユニットを%d個内蔵しています。¥n", info.units);
    printf("同時に振動設定が可能なユニットは%d個です。¥n", info.se_units);
} else {
    /* 振動デバイスは接続されていない */
}
```

同時設定可能なユニット数とは、1 回のコマンド発行 ( 1 回の `pdVibStart()` 関数呼び出し ) で振動可能なユニット数です。

### 3.2.4 振動ユニット情報の取得

関数 `pdVibGetInfo` で、デバイスの持つすべてのユニットの情報を取得します。このユニット情報を取得するにはデバイスとの通信が必要なため、1 ユニットあたり期間の時間が必要です。次のコードを参考にしてください。

```

    Uint32 flag;

    {
        case PDD_VIBERR_NO_VIBRATOR:
            printf("                ¥n");
            flag = TRUE;                コールバックはこれ以降発生しない */

        case PDD_VIBERR_OK:
            printf("                %d の情報
            printf("位置
                :
                :
                /* 全ユニットのコールバック終了
            break;

    }

}

-----

flag = FALSE;

if (pdVibGetInfo(PDD_PORT_A2, &info) != PDD_VIBERR_OK) return NG;

PDD_VIBERR_OK) return NG;

while (!flag);                全ユニットのコールバックを待つ */

```

これにより、関数 `pdVibGetInfo` がユニットの個数回呼び出され、次々とユニット情報を取得できます。注意すべき点は、関数 `pdVibGetInfo` 自体は即時復帰であり、コールバックは1回に1回ずつ発生するという点です。

### 3.2.5 振動ユニット情報の詳細

取得したユニット情報構造体 PDS\_VIBUNITINFO の詳細は以下のようになっています。

メンバ		値	意味
	振動ユニットの位置を示します。		デバイスの前部に位置しています
		PDD_VIB_POS_BACK	
		PDD_VIB_POS_LEFT	デバイスの左側に位置しています
axis	振動軸を示します。		デバイスの右側に位置しています
		PDD_VIB_AXIS_NONE	軸方向なし(回転運動等)
		PDD_VIB_AXIS_X	X 軸方向(プレイヤーに対して左右方向)
		PDD_VIB_AXIS_Y	Y 軸方向(プレイヤーに対して上下方向)
pow_enable	振動強度が可変かどうかを示します。		Z 軸方向(プレイヤーに対して前後方向)
		PDD_VIB_AXIS_Z	
dir_enable	振動方向の指定が可能かどうかを示します。	0	固定
		1	8段階で可変
wave_enable	任意の振動波形が設定可能かどうかを示します。	0	不可(振動方向は固定)
		1	可(+、-で正方向、逆方向指定可能)
min_freq	設定可能な最小振動周波数	0	不可
max_freq	設定可能な最大振動周波数	0	可

表 3-2.振動ユニット情報

### 3.3 情報取得に関する関数の説明

関数	機能	
pdVibIsReady	振動デバイスの接続を調べる	
pdVibGetInfo	振動デバイスの情報を取得する	
pdVibEnumerateUnit	振動デバイスの全振動ユニットの情報を取得する	
pdVibGetDirection	振動デバイスの接続されている向きを取得する	
PDS_VIBENUMUNITCALLBACK	コールバック関数型	

表 3-3.関数一覧

構造体	機能
PDS_VIBINFO	振動デバイスの情報を格納する
PDS_VIBUNITINFO	振動ユニットの情報を格納する

表 3-4.構造体一覧

#### •API

#### pdVibIsReady

振動デバイス関数

書式 Sint32 pdVibIsReady(UINT32 port)

パラメータ port ポート番号 PDD\_PORT\_A1/A2/B1/B2/C1/C2/D1/D2

戻り値 TRUE 振動デバイスが接続されている  
FALSE 振動デバイスが接続されていない

機能 ・振動デバイスが接続されているかどうかを調べます

参照

備考

例 if (pdVibIsReady(PDD\_PORT\_A2)) return TRUE;

**pdVibGetDirection**

情報取得関数

書式	Sint32 pdVibGetDirection(Uint32 port)
パラメータ	port                      ポート番号                      PDD_PORT_A1/A2/B1/B2/C1/C2/D1/D2
戻り値	PDD_VIB_DIRECTION_NORMAL                      通常 PDD_VIB_DIRECTION_FLIP                      逆さま PDD_VIB_DIRECTION_LEFT                      左向き PDD_VIB_DIRECTION_RIGHT                      右向き
機能	・振動デバイスの接続されている向きを取得します。
参照	
備考	・接続されている向きにより、振動軸そのものの向きが変化します。
例	

**pdVibGetInfo**

情報取得関数

書式	Sint32 pdVibGetInfo(Uint32 port, PDS_VIBINFO* vibinfo)
パラメータ	port                      ポート番号                      PDD_PORT_A1/A2/B1/B2/C1/C2/D1/D2 vibinfo                      振動デバイス情報を格納するアドレス
戻り値	PDD_VIBERR_OK                      正常終了 PDD_VIBERR_NO_VIBRATOR                      振動デバイスは接続されていない
機能	・振動デバイスの情報を取得します。
参照	pdVibEnumerateUnit(), PDS_VIBINFO、PDS_VIBUNITINFO
備考	
例	<pre> PDD_VIBINFO info; if (pdVibGetInfo(PDD_PORT_A2, &amp;info) == PDD_VIBERR_OK) {     /* 成功 */ } </pre>

**pdVibEnumerateUnit**

情報取得関数

書式	Sint32 pdVibEnumerateUnit(Sint32 port, PD_VIBENUMUNITCALLBACK func, Uint32 param)
パラメータ	port                      ポート番号                      PDD_PORT_A1/A2/B1/B2/C1/C2/D1/D2 func                      コールバック関数アドレス param                      コールバック関数パラメータ
戻り値	PDD_VIBERR_OK                      正常終了 PDD_VIBERR_NO_VIBRATOR                      振動デバイスは接続されていない
機能	・振動デバイスの持つユニットの情報をすべて取得します。
参照	pdVibGetInfo(), PD_VIBENUMUNITCALLBACK、PDS_VIBINFO、PDS_VIBUNITINFO
備考	
例	<pre> Sint32 enumfunc(Uint32 unit, Sint32 stat,                 const PDS_VIBUNITINFO* info, Uint32 param) {     return PDD_VIBRET_OK; } ----- if (pdVibEnumerateUnit(PDD_PORT_A2, enumfunc, 0) == PDD_VIBERR_OK) {     /* 成功 */ } </pre>

**PD\_VIBENUMUNITCALLBACK**

コールバック関数型

書式	typedef Sint32 (*PD_VIBENUMUNITCALLBACK)(Uint32 unit, Sint32 stat, const PDS_VIBUNITINFO* info, Uint32 param)
----	---

パラメータ	unit	情報を取得したユニット番号	
	stat	コールバックステータス PDD_VIBERR_OK PDD_VIBERR_NO_VIBRATOR	成功 振動デバイスは 接続されていない
戻り値	info	ユニット情報を格納したアドレス	
	param	コールバックパラメータ	
機能	PDD_VIBRET_OK を返してください。		
	<ul style="list-style-type: none"> <li>・pdVibEnumerateUnit()関数を使用した際に、1ユニットの情報取得完了毎に呼び出されます。</li> <li>・振動デバイスの持つユニット数と同じ回数コールバックが発生します。</li> <li>・ユニット情報の取得に成功した場合、コールバックステータスは PDD_VIBERR_OK となります。</li> <li>・途中で振動デバイスが取り外された場合、コールバックステータスは PDD_VIBERR_NO_VIBRATOR となります。それ以降コールバックは発生しません。</li> </ul>		
参照 備考 例	pdVibEnumerateUnit(), PDS_VIBUNITINFO		
	<pre>Sint32 enumfunc(Uint32 unit, Sint32 stat,                 const PDS_VIBUNITINFO* info, Uint32 param) {     if (stat == PDD_VIBERR_OK) {         printf("ユニット%dの情報\n", unit);         ;     }     return PDD_VIBRET_OK; }</pre>		

## ・構造体

**PDS\_VIBINFO**

構造体

定義	typedef struct { Uint8 units; Uint8 se_units; } PDS_VIBINFO;
説明	振動デバイスの情報を格納する構造体です。
メンバ	units                    ユニット数 se_units                同時に振動設定の可能なユニット数
参照	pdVibGetInfo(), PDS_VIBUNITINFO

**PDS\_VIBUNITINFO**

構造体

定義	typedef struct { Uint8 position; Uint8 axis; Uint8 pow_enable; Uint8 cont_enable; Uint8 dir_enable; Uint8 wave_enable; Uint8 min_freq; Uint8 max_freq; } PDS_VIBUNITINFO;
説明	振動ユニットの情報を格納する構造体です
メンバ	position                振動ユニットの位置 axis                    振動軸 pow_enable              強度が可変かどうか

	cont_enable	連続振動が可能かどうか
	dir_enable	方向指定が可能かどうか
	wave_enable	任意波形設定が可能かどうか
	min_freq	最小振動周波数
	max_freq	最大振動周波数
参照	pdVibEnumerateUnit()	PD_VIBENUMUNITCALLBACK

### 3.4 振動させる

振動させるには、関数 `pdVibStart()` を使用します。この時、振動させるためのパラメータを構造体にセットする必要があります。関数 `pdVibEnumerateUnit()` で取得したユニット情報を参照して、適切にパラメータを指定してください。

```
Sint32 ret;
PDS_VIBPARAM param;

param.unit = 1; /* ユニット番号1 */
param.flag = PDD_VIB_FLAG_CONTINUOUS; /* 連続振動あり */
param.power = 7; /* 正方向に 7 の強さ(最大)で */
param.freq = 20; /* 20Hz で */
param.inc = 0; /* 収束、発散なし */

ret = pdVibStart(PDD_PORT_A2, &param, 1); /* 振動させる */

if (ret != PDD_VIBERR_OK) {
    /* 振動させることができなかった */
}
```

上の例では、ユニット1のみを振動させています。複数のユニットを振動させるには次のようにパラメータを構造体の配列として宣言し、使用します。

```
Sint32 ret;
PDS_VIBPARAM param[2];

param[0].unit = 1; /* ユニット番号1 */
param[0].flag = PDD_VIB_FLAG_CONTINUOUS; /* 連続振動あり */
param[0].power = 7; /* 正方向に 7 の強さ(最大)で */
param[0].freq = 20; /* 20Hz で */
param[0].inc = 0; /* 収束、発散なし(0Hz) */

param[1].unit = 2; /* ユニット番号2 */
param[1].flag = PDD_VIB_FLAG_EXHALATION; /* 単発発散振動 */
param[1].power = -7; /* 負方向に 7 の強さ(最大)で */
param[1].freq = 15; /* 15Hz で */
param[1].inc = 1; /* 発散周期 1 */

ret = pdVibStart(PDD_PORT_A2, param, 2); /* 振動させる */

if (ret != PDD_VIBERR_OK) {
    /* 振動させることができなかった */
}
```

`pdVibStart()` 関数の第3パラメータは振動パラメータの個数、すなわち同時に振動設定をしたいユニットの個数です。これは振動デバイス情報にある「同時に振動設定可能なユニット数」を超えることはできません。



3.5 振動に必要なパラメータと設定例

pdVibStart()に渡す 構造体の各メンバの意味と、設定例を解説します。  
各振動ユニットへのパラメータを設定する際には、関数 で取得したユニット情報  
を参照し、適切な値を必ず設定してください。パラメータが不正の場合、そのユニットあるいはデバ  
を起こす場合があります。ライブラリではパラメータのチェックは行いません。

メンバ		指定する値	説明
	振動ユニット番号	1 V.se_units	振動パラメータを送る振動ユニットを指定します。
flag	振動フラグ		連続振動を指定します。 振動自動停止時間が来るまで振動を続けます。 (ワンショット)となります。 U.cont_enable==0
		PDD_VIB_FLAG_EXHALATION	振動がだんだん強くなるよう指定します。 (動) PDD_VIB_FLAG_CONVERGENCE できません。 連続振動時には、最大強度に達すると再び で指定された強度から発散振動を開始します。 U.pow_enable==0
			振動がだんだん弱くなるよう指定します。(収束振 PDD_VIB_FLAG_EXHALATION とは同時に指定でき 連続振動時には、最小強度に達すると再び power U.pow_enable==0 の場合は指定不可です。
	振動の強さ	-7 0 ~	正で正方向、負で逆方向に振動します。 値が大きいほど強い振動となります。 の場合は無視されます。 U.dir_enable==0 は符号は無視されます。
	振動数( 振動周波	U.min_freq U.max_freq	( 振動周波数)を指定します。
inc		1 255	収束、発散させない場合は無視されます。

表 パラメーター一覧

表中の V\_ は 構造体を示します。  
表中の U\_ は 構造体を示します。

### 3.6 振動を止める

振動を止める場合も、関数 VibStart()を使用します。次のようにします。

```
Sint32 ret;
PDS_VIBPARAM param;

param.unit = 2;                /* ユニット番号 2 */
param.flag = 0;                /* フラグなし */
param.power = 0;               /* 振動を止める */
param.freq = 0;                /* 0Hz で */
param.inc = 0;                 /* 収束、発散なし */

ret = pdVibStart(PDD_PORT_A2, &param, 1); /* 振動を止める */

if (ret != PDD_VIBERR_OK) {
    /* 止められなかった */
}
```

上の例では、ユニット2のみを停止させています。複数のユニットを振動させるには、振動させる場合と同様に、パラメータを構造体の配列として宣言し、振動強度を0に設定してください。

### 3.7 自動停止時間の設定

自動停止時間を設定するには、関数 pdVibSetStopTime()を使用します。

```
Sint32 ret;
Uint8 units[] = {1, 3, 5};    /* 設定したいユニット番号を格納した配列 */
Uint8 times[] = {10, 20, 30}; /* ユニット毎の設定値を格納した配列 */

ret = pdVibSetStopTime(PDD_PORT_A2, units, times, 3); /* 設定 */
if (ret != PDD_VIBERR_OK) {
    /* 設定できなかった */
}
```

この例では、ユニット 1、3、5 の自動停止時間をそれぞれ 10 (約 2.5 秒)、20 (約 5 秒)、30 (約 7.5 秒) に設定しています。

## 3.8 振動に関する関数の説明

関数	機能
pdVibStart	振動を開始する
pdVibSetStopTime	振動自動停止時間を設定する

表 3-6.関数一覧

## ・API

pdVibStart	振動デバイス関数
書式	Sint32 pdVibStart(Uint32 port, const PDS_VIBPARAM* param, Uint32 num)
パラメータ	port                   ポート番号       PDD_PORT_A1/A2/B1/B2/C1/C2/D1/D2 param               振動パラメータ構造体配列のアドレス num                  ユニット数
戻り値	PDD_VIBERR_OK           正常終了 PDD_VIBERR_NO_VIBRATOR   振動デバイスは接続されていない PDD_VIBERR_BUSY          振動デバイスは BUSY 状態
機能	・振動デバイスに、振動を開始させます。
参照	PDS_VIBPARAM
備考	・
例	<pre> PDS_VIBPARAM param;  param.unit = 1; param.flag = PDD_VIB_FLAG_CONTINUOUS; param.power = -7; param.freq = 15; param.inc = 0;  if (pdVibMxStart(PDD_PORT_A2, &amp;param) == PDD_VIBERR_OK) {     /* 成功 */ } </pre>

pdVibSetStopTime	振動デバイス関数
書式	Sint32 pdVibSetStopTime(Uint32 port, Uint8* unit, Uint8* time, Uint32 num)
パラメータ	port                   ポート番号       PDD_PORT_A1/A2/B1/B2/C1/C2/D1/D2 unit                  ユニット番号を格納した配列 time                 自動停止時間を格納した配列 num                  ユニット数
戻り値	PDD_VIBERR_OK           正常終了 PDD_VIBERR_NO_VIBRATOR   振動デバイスは接続されていない PDD_VIBERR_BUSY          振動デバイスは BUSY 状態
機能	・振動デバイスの振動自動停止時間を設定します。
参照	・
備考	・
例	<pre> Uint8 units[] = {1, 3, 5}; Uint8 times[] = {10, 20, 30}; if (pdVibSetStopTime(PDD_PORT_A2, units, times, 3) == PDD_VIBERR_OK) {     /* 成功 */ } </pre>

## 第4章 任意振動波形

本バージョンではサポートされていません。