

CPU Control

目次

1. 概要	3
2. 関数リファレンス.....	3
EnableInterrupt	4
DisableInterrupt.....	5
RestoreInterrupt.....	6
3. 仕組み	7
4. 使い方	7
更新履歴	9

1. 概要

本モジュールは、CPU 制御を受け持つモジュールです。

2. 関数リファレンス

本モジュールには、Table2-1.の関数が含まれています。

Table2-1. 関数一覧

関数名	概要
EnableInterrupt	割り込み許可
DisableInterrupt	割り込み禁止
RestoreInterrupt	割り込み状態の復元

EnableInterrupt

書式 :

```
int EnableInterrupt( void );
```

機能 :

割り込み許可

引数 :

なし

返値 :

直前の割り込み許可状態

詳細 :

無条件に割り込みを許可する。返値として以前の割り込み許可状態を返す。

本関数の返値を [RestoreInterrupt](#) 以外で利用すると環境依存のコードになることに注意。直前の状態への復帰は [RestoreInterrupt](#) を使う。

参考 :

[DisableInterrupt](#), [RestoreInterrupt](#)

DisableInterrupt

書式：

```
int DisableInterrupt( void );
```

機能：

割り込みを禁止する

引数：

なし

返値：

直前の割り込み許可状態

詳細：

無条件に割り込みを禁止する。その他は [EnableInterrupt](#) と同じである。

参考：

[EnableInterrupt](#), [RestoreInterrupt](#)

RestoreInterrupt

書式 :

```
void RestoreInterrupt( int RestoreState );
```

機能 :

割り込み許可状態の復元

引数 :

RestoreState ... 割り込み許可状態

返値 :

なし

詳細 :

RestoreState には、[EnableInterrupt](#) か [DisableInterrupt](#) の返値を指定する。それ以外の値を指定した場合の動作は未定義です。

参考 :

[EnableInterrupt](#), [DisableInterrupt](#)

3. 仕組み

[EnableInterrupt](#), [DisableInterrupt](#) を実施すると、割り込み状態を示すフラグレジスタを読み取った後、割り込み許可・禁止を設定します。このとき、割り込み状態を示すフラグレジスタにその他のフラグが含まれていたとしてもマスク等は実施しません。

[RestoreInterrupt](#) を実施すると、フラグレジスタの内容を指定値で更新だけです。ただし、割り込み許可・禁止以外のフラグの更新が悪影響を及ぼす環境の場合に限り、適切なマスク処理が実施されます。

ヘッダファイルは汎用ですが、実体は CPU 個別に異なるので注意してください。

4. 使い方

通常は、`DisableInterrupt()` と `RestoreInterrupt()` のペアで使います。このペアで使うことで、割り込み禁止ブロックのネストが可能になります。

たとえば、下記のようなケースを想定してください。

```
void func1( void ) {
    int State;
    State = DisableInterrupt();
    www();
    func2();
    aaa();
    RestoreInterrupt( State );
}
```

```
void func2( void ) {
    int State;
    State = DisableInterrupt();
    xxx();
    yyy();
    zzz();
    RestoreInterrupt( State );
}
```

func2() は割り込み禁止にしない別の場所からも呼ばれるケースがあり、xxx() ~ zzz()の連続呼び出しは、割り込み禁止でなければなりません。一方で、func1() の中身も www() ~ aaa() の呼び出しは割り込み禁止でなければならないとします。これをもし、RestoreInterrupt() の代わりに EnableInterrput() を実施してしまうと、func1() の中身は、func2() を呼び出した時点で割り込み許可になってしまい、aaa() 呼び出しの直前に割り込まれてしまう可能性があり、目的を満たしません。

DisableInterrupt() と RestoreInterrupt() のペアで使用するにより、それらに挟まれた領域が、すでに割り込み禁止状態で呼ばれた場合でも、そうでなかった場合でも、適切に元通りの割り込み禁止/許可状態に戻されます。

更新履歴

更新日	リビジョン	更新内容	更新者
2004/07/19	1.0.0	初版	t.hara
2004/08/15	1.0.1	引数・返値の型を修正, 使い方を追加	t.hara