encoder

インクリメンタル式ロータリーエンコーダのパルスをカウントするクラス.

struct Encoder_data

エンコーダからの取得値を格納する関数。タイマー割り込みと合わせるて使う。

[要素]
カウンタ値
回転数[回転]
角度[°]
移動距離[mm]
速度[mm/s]
回転速度[rps]

[要素一覧]

```
struct Encoder_data{
    int count;
    double rot,deg,distance,volcity,rps;
};
```

class Encoder

void Encoder::init(Pin a_pin,Pin b_pin,TimerNumber tim_num)

エンコーダのピンとタイマー番号を設定する.
使用することができるタイマーは1,2,3,4,5,8のチャンネル1とチャンネル2である.
エンコーダのカウントに使用するタイマーはチャンネル3とチャンネル4も含めてPWM出力に使用することはできない.

[パラメータ]
チャンネル1のピン番号
チャンネル2のピン番号
タイマー番号

[戻り値]
なし

[サンプルコード]
タイマー2をエンコーダモードに設定する

```
#include "stm32f4xx.h"
#include "sken_library/include.h"

Encoder encoder;

int main(void)
```

```
{
    sken_system.init();
    encoder.init(A0,A1,TIMER2);
    while(1)
    {

    }
}
```

int Encoder::read(void)

カウンタの値を取得することができる.
カウンタの値は-30000から30000までである.

[パラメータ]
なし

[戻り値]
カウンタ値

[サンプルコード]

```
#include "stm32f4xx.h"
#include "sken_library/include.h"

Encoder encoder;

int count;

int main(void)
{
    sken_system.init();
    encoder.init(A0,A1,TIMER2);
    while(1)
    {
        count = encoder.read();
    }
}
```

void Encoder::interrupt(Encoder_data* encoder_data)

Encoder_data構造体にエンコーダからの取得値を格納する関数。タイマー割り込み関数内で使用する。

[パラメータ]
Encoder_data構造体アドレス

[戻り値]
なし

[サンプルコード]
タイマー割り込みで取得値を格納する

```
#include "stm32f4xx.h"
```

```
#include "sken_library/include.h"

Encoder encoder;
Encoder_data e_data;

void encoder_interrupt(){
    encoder.interrupt(&e_data);
}

int main(void){
    sken_system.init();
    encoder.init(A0,A1,TIMER2);
    sken_system.addTimerInterruptFunc(encoder_interrupt,0,1);
    while(1)
    {

    }
}
```

void Encoder::reset(void)

カウンタの値をリセットする

[パラメータ]
なし

[戻り値]
なし

[サンプルコード]
カウンタの値が10000を超えた時にリセットする

```
#include "stm32f4xx.h"
#include "sken_library/include.h"

Encoder encoder;

int count;

int main(void)
{
    sken_system.init();
    encoder.init(A0,A1,TIMER2);
    while(1)
    {
        count = encoder.read();
        if(count > 10000){
            encoder.reset();
        }
    }
}
```