system

millisなどの基本的な関数やCAN通信の制御など

class SYSTEM

void SYSTEM::init(bool use_external_crystal = false)

初期化関数でプログラムで最初に呼び出す必要がある.
sken_systemという名前ですでにオブジェクトが用意されている

[パラメータ]
外部クロック使用

[戻り値]
なし

[サンプルコード]

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

int main(void)
{
    sken_system.init();
    while(1)
    {

    }
}
```

unsigned int SYSTEM::micros(void)

マイコン起動時からの時間をマイクロ秒で返す.

[パラメータ]
なし

[戻り値]
マイコン起動時からの時間

[サンプルコード]
マイコン起動時からの時間を取得する.

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

unsigned int time;

int main(void)
{
    sken_system.init();
    while(1)
    {
        time = sken_system.micros();
    }
}
```

unsigned int SYSTEM::millis(void)

マイコン起動時からの時間をミリ秒で返す.

[パラメータ]
なし

[戻り値]
マイコン起動時からの時間

[サンプルコード]
マイコン起動時からの時間を取得する.

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

unsigned int time;

int main(void)
{
    sken_system.init();
    while(1)
    {
        time = sken_system.millis();
    }
}
```

void SYSTEM::delayMillis(unsigned int delay_time_millis)

プログラムを指定した時間だけ停止させる.
単位はミリ秒.

[パラメータ]
停止時間

[戻り値]
なし

[サンプルコード]
1秒待ってからループに入る.

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

int main(void)
{
    sken_system.init();
    sken_system.delayMillis(1000);
    while(1)
    {

    }
}
```

void SYSTEM::delayMicros(unsigned int delay_time_micros)

プログラムを指定した時間だけ停止させる.
単位はマイクロ秒.

[パラメータ]
停止時間

[戻り値]
なし

[サンプルコード]
1ミリ秒待ってからループに入る.

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

int main(void)
{
    sken_system.init();
    sken_system.delayMicros(1000);
    while(1)
    {

    }
}
```

bool SYSTEM::addTimerInterruptFunc(void(*function_p)(void),int id,int period = 1)

タイマー割り込み関数を追加する.

[パラメータ]
タイマー割り込み時に呼び出す関数のアドレス.
関数に与えるID. 0から7の間で選択する.
関数の呼び出し周期. デフォルトは1[ms]である.

[戻り値]
成功したらtrue, 失敗したらfalse.

[サンプルコード]
1[ms]ごとにカウンタをインクリメントする.

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

unsigned int counter;

void func(void)
{
    counter++;
}

int main(void)
{
    sken_system.init();
    sken_system.addTimerInterruptFunc(func,0,1);
    while(1)
    {

    }
}
```

bool SYSTEM::deleteTimerInterruptFunc(int id)

指定したIDのタイマー割り込み関数を削除する.

[パラメータ]
削除する関数のID. 0から7の間で選択する.

[戻り値]
成功したらtrue, 失敗したらfalse

[サンプルコード]
1[ms]ごとにカウンタをインクリメントし, 1000を超えたら停止する.

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

unsigned int counter;

void func(void)
{
    counter++;
}

int main(void)
{
    sken_system.init();
    sken_system.addTimerInterruptFunc(func,0,1);
    while(1)
    {
        if(counter > 1000){
            deleteTimerInterruptFunc(0);
        }
    }
}
```

bool SYSTEM::changeTimerInterruptPeriod(int id,int period)

指定したIDのタイマー割り込み関数の周期を変更する．

[パラメータ]
周期を変更する関数のID．0から7の間で選択する．

[戻り値]
成功したらtrue，失敗したらfalse

[サンプルコード]
1[ms]ごとにカウンタをインクリメントし，1000を超えたら2[ms]ごとにカウンタをインクリメントする．

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

unsigned int counter;

void func(void)
{
    counter++;
}

int main(void)
{
    sken_system.init();
    sken_system.addTimerInterruptFunc(func,0,1);
    while(1)
    {
        if(counter > 1000){
            changeTimerInterruptPeriod(0,2);
        }
    }
}
```

bool SYSTEM::startCanCommunicate(PIN tx_pin,PIN rx_pin,CAN_SELECT can_select)

CAN通信を開始する．
通信速度は1[Mbps]である．

[パラメータ]
CAN_TXピン
CAN_RXピン
CAN番号．CAN_1かCAN_2から選択．

[戻り値]
成功したらtrue，失敗したらfalse

[サンプルコード]
CAN_1の通信を開始する．

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

int main(void)
{
    sken_system.init();
    sken_system.startCanCommunicate(B9,B8,CAN_1);
    while(1)
    {

    }
}
```

bool SYSTEM::canTrancemit(CAN_SELECT can_select,uint32_t stdid,uint8_t* data_p,int data_size,int dead_time = 10)

CAN通信で送信する.

[パラメータ]
CAN番号．CAN_1かCAN_2から選択.
CAN通信におけるスタンダードID.
送信データアドレス.
送信データ数.
デッドタイム．デフォルトは10[ms]である.

[戻り値]
成功したらtrue，失敗したらfalse

[サンプルコード]
CAN_1でデータを送信する.

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

uint8_t can_transmit_data_1[8] = {};

int main(void)
{
    sken_system.init();
    sken_system.startCanCommunicate(B9,B8,CAN_1);
    while(1)
    {
        sken_system.canTrancemit(CAN_1,0x1010,can_transmit_data_1,8);
    }
}
```

bool SYSTEM::addCanRceiveInterruptFunc(CAN_SELECT can_select,void(*function_p)(CanRxMsgTypeDef),int id)

CAN通信で受信割り込み関数を追加する.

[パラメータ]
CAN番号．CAN_1かCAN_2から選択．
受信割り込み時に呼び出す関数のアドレス．
関数に与えるID．0から7の間で選択する．

[戻り値]
成功したらtrue，失敗したらfalse

[サンプルコード]
CAN_1の受信割り込み関数を追加する．

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

void func(CanRxMsgTypeDef)
{

}

int main(void)
{
    sken_system.init();
    sken_system.startCanCommunicate(B9,B8,CAN_1);
    sken_system.addCanRceiveInterruptFunc(CAN_1,func,0);
    while(1)
    {

    }
}
```

bool SYSTEM::deleteCanRceiveInterruptFunc(CAN_SELECT can_select,int id)

CAN通信で受信割り込み関数を削除する．

[パラメータ]
CAN番号．CAN_1かCAN_2から選択．
削除する関数のID．0から7の間で選択する．

[戻り値]
成功したらtrue，失敗したらfalse

[サンプルコード]
CAN_1の受信割り込み関数を1秒後に削除する．

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

void func(CanRxMsgTypeDef)
{

}

int main(void)
{
```

```
    sken_system.init();
    sken_system.startCanCommunicate(B9,B8,CAN_1);
    sken_system.addCanRceiveInterruptFunc(CAN_1,func,0);
    sken_system.delayMillis(1000);
    sken_system.deleteCanRceiveInterruptFunc(CAN_1,0);
    while(1)
    {

    }
}
```