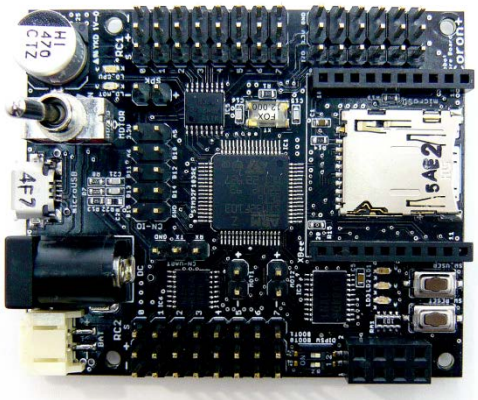
Coron+

Robot Core Board

コロン・コロンプラス共通
ソフトウェアライブラリ

v3.0.0 版



TECHNO ROAD Inc.

http://techno-road.com/

v3.0.0　2015/9/7

※　Coron+では、v3.0.0（本書）以降を使用して下さい。
　また、従来の Coron（無印）でも v3.0.0 を使用できます。

～目次～

Coron ライブラリについて

Coron 及び Coron＋（以下まとめて「Coron」と記載）の開発環境には，大変多くのライブラリが用意されており，大きく分けて以下の３つに分類されます.

① Coron 専用ライブラリ(coron_○○○.h)
Coron ボードに実装されている RC サーボや DC モータ，スピーカ，microSD カード用の簡単な FAT16 システムライブラリが用意されているパッケージ.

② 標準ファームウェアライブラリ(stm32f10x_○○○.h …etc)
CPU の割り込みやクロックの設定，GPIO，UART，A/D といった周辺機能などの基本的な機能の設定が用意されているパッケージ.

③ USB ライブラリ(usb_○○○.h)
USB 機能を利用する際に使うライブラリ.
CPU に内蔵されている USB モジュールのコアライブラリとその API が用意されているパッケージ．USB の仮想 COM ポートや，HID デバイス，マスストレージなど USB の基本機能を実現する.

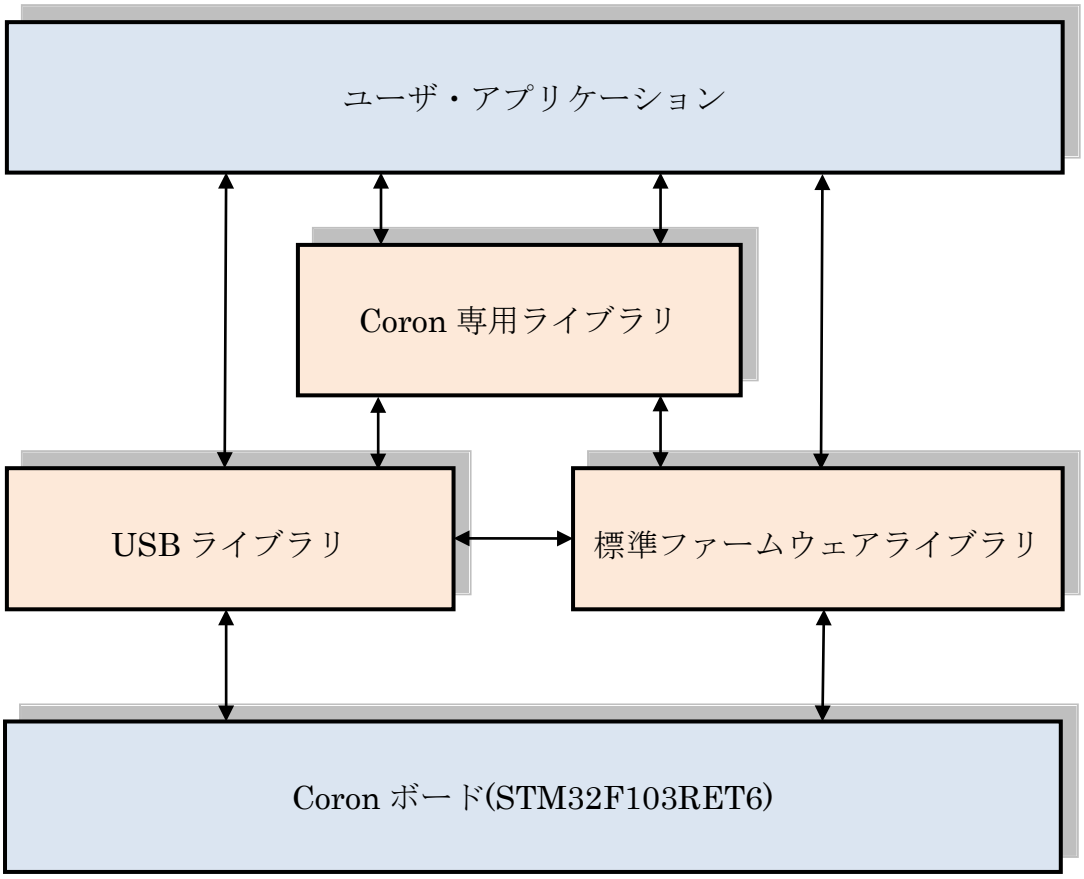


図　Coron ライブラリシステム階層図

1. Coron 専用ライブラリ

Coron 初期化関数、スイッチ＆LED 命令（Coron_init.h）

関数名	void **coron_init**()
機能	メインクロックの初期化，LD1・LD2・LD3 と SW_USER の初期化，SysTick の初期化をする (IDE3より、スタートアップルーチンから自動的に実行される)
命令名	**SW_USER**
機能	プッシュスイッチ(SW_USER)の状態を取得（※押す=1，離す=0）
命令名	**LD1_OFF**
機能	LD1（緑 LED）を消灯する
命令名	**LD1_ON**
機能	LD1（緑 LED）を点灯する
命令名	**LD2_OFF**
機能	LD2（橙 LED）を消灯する
命令名	**LD2_ON**
機能	LD2（橙 LED）を点灯する
命令名	**LD3_OFF**
機能	LD3（橙 LED）を消灯する
命令名	**LD3_ON**
機能	LD3（橙 LED）を点灯する

```
/*** プログラム例 ***/
#include "stm32f10x_lib.h"                    //標準ファームウェアライブラリ
#include "coron.h"
#include "coron_init.h"
void main(void){
      while(1){
          if(SW_USER==1){                     //スイッチが押されたら
                LD1_ON;                       //LD1 を点灯
          }
          else if(SW_USER==0){ //スイッチが離されたら
                LD1_OFF;                      // LD1 を消灯
          }
     }
}//end
```

Coron 待機関数（Coron_wait.h）

関数名	void **wait_timer**(u32 time_x100us)
機能	指定された時間（x100[usec]）だけ待機する. SysTick 割り込みを使用
関数名	void **wait_timer_sec**(u32 time_sec)
機能	指定された時間[sec]だけ待機する. SysTick 割り込みを使用
関数名	void **wait_timer_msec**(u32 timer_msec)
機能	指定された時間[msec]だけ待機する. SysTick 割り込みを使用
関数名	void **wait_timer_usec**(u32 timer_usec)
機能	指定された時間[usec]だけ待機する. SysTick 割り込みを使用
関数名	void **wait_loop**(vu32 loop)
機能	指定された値の数だけ減算ループして時間を稼ぐ
関数名	void **wait_init**(void)
機能	タイマの初期化をする

Coron RC サーボ出力関連（Coron_rc.h）

関数名	void **RC_init**(void)
機能	RC サーボ出力の初期化をする(※初期化完了後は出力禁止状態)
関数名	void **RC_enable**(void)
機能	RC サーボ出力許可する
関数名	void **RC_disable**(void)
機能	RC サーボ出力禁止する
関数名	void **RC_exeHomePosition**(void)
機能	ホームポジションをとる
関数名	void **RC_move**(u8 time)
機能	指定された時間(x20[msec])で現在の出力角から目標角(rc_mot_ipos[ ][ ])まで移動する. 動作終了時に rc_step_flag が 0 になる

グローバル変数名	vu16 **rc_mot_pos**[2][8]
説明	各 RC サーボの出力実行角（PWM_High 時間[usec]） rc_mot_pos[ポート][番号] = 700～2300
グローバル変数名	vu16 **rc_mot_ipos**[2][8]
説明	各 RC サーボの目標角（PWM_High 時間[usec]） rc_mot_ipos[ポート][番号] = 700～2300
グローバル変数名	vu16 **rc_mot_pos_home**[2][8]
説明	各RC サーボのホームポジション（PWM_High 時間[usec]） rc_mot_pos[ポート][番号] = 700～2300
グローバル変数名	vs16 **rc_mot_pos_trim**[2][8]
説明	各RC サーボのトリム（PWM_High 時間[usec]） rc_mot_pos[ポート][番号] = 700～2300
グローバル変数名	vu8 **rc_step_flag**
説明	RC_move()の終了確認フラグ 動作時＝1, 終了時＝0

```
/*** プログラム例 ***/
 rc_mot_pos[0][0]=1500;      //RC0-0 のサーボを中心角 0°（1500[usec])に移動する
 wait_timer(10000);          //100[usec] x (10000) = 1[sec]待つ
 rc_mot_ipos[0][0]=2300;     // RC0-0 の目標角を+90°（2300[usec])にセットする
 RC_move(100);               //20[msec] x (100) = 2[sec]かけて目標角に移動する
 while(rc_step_flag);        //移動し終わるまで待つ
```

Coron DC モータ出力関連（Coron_dc.h）

関数名	void **DC_init**(void)
機能	DC モータ出力の初期化をする(※初期化完了後は出力禁止状態)
関数名	void **DC_enable**(void)
機能	DC モータ出力許可する
関数名	void **DC_disable**(void)
機能	DC モータ出力禁止する

グローバル変数名	vs8 **mot1_cw**, **mot2_cw**
説明	MOT1,MOT2 の回転モード MODE_CW :正転 MODE_CCW :逆転 MODE_BRAKE :ブレーキ MODE_STOP :ストップ(ハイインピーダンス)
グローバル変数名	vu8 **mot1_duty**, **mot2_duty**
説明	MOT1,MOT2 の PWM 出力 Duty 比[%]

```
/*** プログラム例 ***/
   DC_init();                  //DC モータ初期化
   mot1_cw=MODE_CW;            //MOT1 の回転モードをセット(正転)
   mot2_cw=MODE_CCW;           //MOT2 の回転モードをセット(逆転)
   mot1_duty=50;               //MOT1 の Duty 比を 50%に設定
   mot2_duty=100;              //MOT2 の Duty 比を 100%に設定
   DC_enable();                //DC モータ出力許可
   while(1);
//end
```

Coron A/D コンバータ関連（Coron_adc.h）

関数名	void **IOA_AD_init**(u8 port)
機能	A/D コンバータを初期化する
関数名	void **AD_enable**(void)
機能	A/D コンバータへの入力を許可する
関数名	void **AD_disable**(void)
機能	A/D コンバータへの入力を禁止する

グローバル変数名	vu16 **coron_IOA_ADValue**[6]
説明	AD 変換格納バッファ

Coron スピーカ出力関連（Coron_sound.h）

関数名	void **SPK_init**(void)
機能	スピーカ出力の初期化をする モノラル，サンプルサイズ：8 ビット，サンプリングレート：22kHz

グローバル変数名	vu8 **sound_buf**[2][512]
説明	サウンド・データ・バッファ 512byte のデータ(x2)が交互に再生される
グローバル変数名	vu8 **sound_flag**
説明	再生フラグ．1 をセットすると再生開始 512byte の再生が終わる度に,1 と 2 が交互にセットされる

Coron WAV ファイル関連（Coron_sdwav.h）

関数名	void **SD_search_wav**(void)
機能	スピーカ出力の初期化をする モノラル，サンプルサイズ：8 ビット，サンプリングレート：22kHz
関数名	int **SD_play_wav**(u8 wav_num)
機能	指定された番号(データ順)の wav ファイルをスピーカで再生する SD_init( )成功後に使用可能
関数名	void **SD_stop_wav**(void)
機能	wav ファイルの再生を停止．SD_init()成功後に使用可能
関数名	int **SD_nowPlaying_wav**(void)
機能	再生中の wav ファイルの番号を取得(0 番目から)

グローバル変数名	u32 **sound_sector**
説明	WAV ファイルディレクトリのセクタ番号
グローバル変数名	u32 **wav_sector_num**[50]
説明	各 WAV ファイルのセクタ番号
グローバル変数名	u32 **wav_size**[50]
説明	WAV ファイルのサイズ
グローバル変数名	char **wav_name**[50][8]
説明	WAV ファイルの名前
グローバル変数名	u8 **wav_count**
説明	WAV ファイルの数

Coron SD カード FAT システム関連（Coron_sdfat.h）

関数名	int **SD_FATinit**(void)
機能	SD カードの初期化をする(SDIO) 成功=1　失敗＝0（SD カードが挿入されていないなど）
関数名	void **SD_read**(u32 adr)
機能	指定したアドレスから 1 セクタ分のデータを読み出す 結果はグローバル変数 buf512[512] に格納される
関数名	void **SD_write**(u32 adr)
機能	グローバル変数 buf512[512] のデータを指定されたアドレスから1セクタ分書き込む
関数名	s16 **SD_char_to_num**(char *s,char len)
機能	ASCII コードの数字(文字列)を数値に変換する
関数名	void **SD_num_to_char**(long num,u8 len,char *s)
機能	数値を ASCII コードの数字(文字列)に変換する
関数名	void **SD_getDirSector**(void)
機能	SD カードの各 FAT セクタを取得する
関数名	u16 **SD_getFileList**(u32 sec, char *sn, char (file_name)[8], u32 *sct_num, u32 *size)
機能	指定されたディレクトリ内の指定された拡張子名のファイル情報取得
関数名	u32 **SD_searchDirectory**(const char* s)
機能	指定された文字列のディレクトリのセクタを検索

グローバル変数名	u32 **fat_sector**, **dir_sector**, **dat_sector**
説明	各 FAT システム・ディレクトリのセクタ番号 SD_FATinit( )実行時に取得される
グローバル変数名	u8 **sectors_per_cluster**
説明	１クラスタあたりのセクタ数
グローバル変数名	u32 **buf512**[512]
説明	１セクタ分のバッファ SD_read( ) SD_write( ) ともにこのバッファのデータを使って処理がされる

SD カード 1 ブロック(512Byte)書込＆読込

```
/*** プログラム例 ***/
  for( i=0;i<512;i++ ){ buf512[i] = 'a';}   //1 ブロック分のデータを'a'でうめる
  if( SD_FATinit ){                         //SD カードを初期化する
      SD_write(100 * BlockSize);            //セクタ番号 100 に 1 ブロック分書き込む
      SD_read(100 * BlockSize);             //セクタ番号 100 を 1 ブロック分読み出す
      if( buf512[511] = 'a' ){              //SD カードに'a'が無事に書き込まれたら
          LD1_ON;                           //   LD 1 を点灯する
       }
  }
  else{}                                    //SD カード初期化失敗
//end
```

SD カード WAV ファイル・スピーカ再生

```
/*** プログラム例 ***/
  SPK_init( );                              //スピーカ出力を初期化する
  if( SD_FATinit ){                         //SD カードを初期化する
      SD_search_wav( );                     //"SoundData"フォルダ内の
                                            //    WAV ファイルの各情報を取得する
                                            //      ファイル名：wav_name[50][8]
                                            //      サイズ     ：wav_size[50]
                                            //      ファイル数：wav_count
      SD_play_wav(0);                       //1 番目の WAV ファイルを再生する
      SD_play_wav(1);                       //2 番目の WAV ファイルを再生する
  }
  else{}                                    //SD カード初期化失敗
//end
```

※microSD カード内に"SoundData"というフォルダを作り，そのフォルダ内に
WAV ファイル(**モノラル**，**サンプルサイズ**：8 **ビット**，**サンプリングレート**：22kHz)を保存する．

Coron SD カードテキスト 入出力関連（Coron_sdtxt.h）

関数名	char **SD_txt_read_c**(u8 txt_num, u32 s_num)
機能	SD カード内テキストから 1 文字読み込み
関数名	void **SD_txt_write_c**(u8 txt_num, u32_s_num, char wbuf)
機能	SD カード内テキストに 1 文字書き込み
関数名	void **SD_txt_read_s**(u8 txt_num, u32 s_num, char *buf, u16 len)
機能	SD カード内テキストから文字列読み込み
関数名	void **SD_txt_write_s**(u8 txt_num, u32 s_num, char* buf, u16 len)
機能	SD カード内テキストに文字列書き込み

Coron UART 入出力関連（Coron_uart.h）

関数名	void **UART_init**(void)
機能	UART を初期化する
関数名	void **UART_putc**(char data)
機能	UART 1 文字出力
関数名	void **UART_puts**(char* buf)
機能	UART 文字列出力
関数名	void **UART_putd**(char* buf, u16 len)
機能	UART 指定された文字数(len)だけ文字列を出力
関数名	void **UART_putn**(s32 num, char len)
機能	UART 指定された数値(num)を文字数分(len)で十進数出力する ex: USB_putn(123,5); -> "00124"
関数名	void **UART_putx**(u32 num, char len)
機能	UART 指定された数値(num)を文字数分(len)で十六進数出力する ex: USB_putx(123,5); -> "0008B"
関数名	signed char **UART_getc**(char wf)
機能	UART 1 文字入力, -1 はデータなし
関数名	u16 **UART_gets**(char* buf)
機能	UART 文字列入力, 戻り値は入力文字数

Coron USB CDC クラス初期化関連（Coron_usbcdc.h）

関数名	void **Set_USBClock**(void)
機能	USB のクロックソース(48MHz)を初期化する
関数名	void **USB_Interrupts_Config**(void)
機能	USB 割り込み処理を初期化する

Coron USB CDC クラス入出力関連（Coron_usbprint.h）

関数名	void **USB_putn**(s32 num, char len)
機能	指定された数値(num)を文字数分(len)で表示する ex: USB_putn(123,5); -> "00123"
関数名	void **USB_putx**(u32 num, char len)
機能	指定された数値(num)を文字数分(len)で十六進数出力する ex: USB_putx(123,5); -> "0007B"
関数名	void **USB_putc**(char data)
機能	一文字送信
関数名	void **USB_puts**(char* buf)
機能	文字列送信
関数名	void **USB_putd**(char* buf, char len)
機能	指定された文字数(len)だけ文字列を出力
関数名	signed char **USB_getc**(char wf)
機能	1 文字入力，-1 はデータなし
関数名	char **USB_gets**(char *buf)
機能	文字列受信
関数名	char **USB_getr**(char *buf, char len)
機能	指定数文字列入力

2. 標準ファームウェアライブラリ

A/D コンバータ関連(stm32f10x_adc.h)

関数名	void **ADC_DeInit**(ADC_TypeDef* ADCx)
機能	ADCx ペリフェラル・レジスタを初期値にリセットする
関数名	void **ADC_Init**(ADC_TypeDef* ADCx , ADC_InitTypeDef* ADC_InitStruct)
機能	ADC_InitStruct で指定されたパラメータに応じて，ADCx ペリフェラルを初期化する
関数名	void **ADC_StructInit**(ADC_InitTypeDef* ADC_InitStruct)
機能	各 ADC_InitStruct メンバに初期値を書き込む
関数名	void **ADC_Cmd**(ADC_TypeDef* ADCx, FunctionalState NewState)
機能	指定された ADC ペリフェラルを有効化/無効化する
関数名	void **ADC_DMACmd**(ADC_TypeDef* ADCx, FunctionalState NewState)
機能	指定された ADC_DMA 要求を有効化/無効化する
関数名	void **ADC_ITConfig**(ADC_TypeDef* ADCx, u16 ADC_IT, FunctionalState NewState)
機能	指定された ADC 割り込みを有効化/無効化する
関数名	void **ADC_ResetCalibration**(ADC_TypeDef* ADCx)
機能	選択された ADC 校正レジスタをリセットする
関数名	FlagStatus **ADC_GetResetCalibrationStatus**(ADC_TypeDef* ADCx)
機能	選択された ADC 校正レジスタのステータスを取得する
関数名	void **ADC_StartCalibration**(ADC_TypeDef* ADCx)
機能	選択された ADC 校正プロセスを開始する
関数名	FlagStatus **ADC_GetCalibrationStatus**(ADC_TypeDef* ADCx)
機能	選択された ADC 校正のステータスを取得する
関数名	void **ADC_SoftwareStartConvCmd**(ADC_TypeDef* ADCx, FunctionalState NewState)
機能	選択された ADC ソフトウェア開始変換を有効化/無効化する
関数名	FlagStatus **ADC_GetSoftwareStartConvStatus**(ADC_TypeDef* ADCx)
機能	選択された ADC ソフトウェア開始変換のステータスを取得する

A/D コンバータ関連(stm32f10x_adc.h)

関数名	void **ADC_DiscModeChannelCountConfig**(ADC_TypeDef* ADCx, u8 Number)
機能	選択された ADC レギュラ・グループ・チャネルに対して不連続モードを構成する
関数名	void **ADC_DiscModeCmd**(ADC_TypeDef* ADCx, FunctionalState NewState)
機能	指定された ADC に対して，レギュラ・グループ・チャネルにおいて不連続モードを有効化/無効化する
関数名	void **ADC_RegularChannelConfig** (ADC_TypeDef* ADCx, u8 ADC_Channel, u8 Rank, u8 ADC_SampleTime)
機能	サンプル時間およびシーケンサ内におけるランクに応じて，選択された ADC レギュラ・チャネルを構成する
関数名	void **ADC_ExternalTrigConvCmd**(ADC_TypeDef* ADCx, FunctionalState NewState)
機能	外部トリガによる ADCx変換を有効化/無効化する
関数名	u16 **ADC_GetConversionValue**(ADC_TypeDef* ADCx)
機能	レギュラ・チャネルに対して，最終 ADCx変換結果を返す
関数名	u32 **ADC_GetDualModeConversionValue**(void)
機能	デュアル・モードにおける最終 ADCx変換結果を返す
関数名	void **ADC_AutoInjectedConvCmd**(ADC_TypeDef* ADCx, FunctionalState NewState)
機能	レギュラ・グループ変換後の，選択された ADC 自動追加グループ変換を有効化/無効化する
関数名	void **ADC_InjectedDiscModeCmd**(ADC_TypeDef* ADCx, FunctionalState NewState)
機能	指定された ADC における追加変換チャネルに対する不連続モードを有効化/無効化する
関数名	void **ADC_ExternalTrigInjectedConvConfig** (ADC_TypeDef* ADCx, u32 ADC_ExternalTrigInjecConv)
機能	追加チャネル変換に対する ADCx外部トリガを構成する
関数名	void **ADC_ExternalTrigInjectedConvCmd** (ADC_TypeDef* ADCx, FunctionalState NewState)
機能	外部トリガによる ADCx追加チャネル変換を有効化/無効化する
関数名	void **ADC_SoftwareStartInjectedConvCmd** (ADC_TypeDef* ADCx, FunctionalState NewState)
機能	選択された追加チャネル変換の ADC 開始を有効化/無効化する
関数名	FlagStatus **ADC_GetSoftwareStartInjectedConvCmdStatus**(ADC_TypeDef* ADCx)
機能	選択された ADC ソフトウェア開始変換ステータスを取得する

A/D コンバータ関連(stm32f10x_adc.h)

関数名	void **ADC_InjectedChannelConfig** (ADC_TypeDef* ADCx, u8 ADC_Channel, u8 Rank, u8 ADC_SampleTime)
機能	サンプル時間およびシーケンサにおけるランクに応じて，選択された ADC 追加チャネルを構成する
関数名	void **ADC_InjectedSequencerLengthConfig**(ADC_TypeDef* ADCx, u8 Length)
機能	追加チャネルに対するシーケンス長を構成する
関数名	void **ADC_SetInjectedOffset** (ADC_TypeDef* ADCx, u8 ADC_InjectedChannel, u16 Offset)
機能	追加チャネル変換値オフセットを設定する
関数名	u16 **ADC_GetInjectedConversionValue** (ADC_TypeDef* ADCx, u8 ADC_InjectedChannel)
機能	選択された追加チャネルに対する ADC 変換結果データを返す
関数名	void **ADC_AnalogWatchdogCmd**(ADC_TypeDef* ADCx, u32 ADC_AnalogWatchdog)
機能	単一/全レギュラ/追加チャネル上のアナログ・ウォッチドッグを有効化/無効化する
関数名	void **ADC_AnalogWatchdogThresholdsConfig** (ADC_TypeDef* ADCx, u16 HighThreshold, u16 LowThreshold)
機能	アナログ・ウォッチドッグの高/低スレッショルドを構成する
関数名	void **ADC_AnalogWatchdogSingleChannelConfig** (ADC_TypeDef* ADCx, u8 ADC_Channel)
機能	アナログ・ウォッチドッグの保護された単一チャネルを構成する
関数名	void **ADC_TempSensorVrefintCmd**(FunctionalState NewState)
機能	温度センサおよび Vrefint チャネルを有効化/無効化する
関数名	FlagStatus **ADC_GetFlagStatus**(ADC_TypeDef* ADCx, u8 ADC_FLAG)
機能	指定された ADC フラグがセットされたか否かを確認する
関数名	void **ADC_ClearFlag**(ADC_TypeDef* ADCx, u8 ADC_FLAG)
機能	ADCx ペンディング・フラグをクリアする
関数名	ITStatus **ADC_GetITStatus**(ADC_TypeDef* ADCx, u16 ADC_IT)
機能	指定された ADC 割り込みが発生したか否かを確認する
関数名	void **ADC_ClearITPendingBit**(ADC_TypeDef* ADCx, u16 ADC_IT)
機能	ADCx割り込みペンディング・ビットをクリアする

DMA 関連(stm32f10x_dma.h)

関数名	void **DMA_DeInit**(DMA_Channel_TypeDef* DMAy_Channelx)
機能	DMA_Channelx レジスタを初期値にリセットする
関数名	void **DMA_Init** (DMA_Channel_TypeDef* DMAy_Channelx, DMA_InitTypeDef* DMA_InitStruct)
機能	DMA_InitStruct 内にて指定されたパラメータに応じて，DMA_Channelx を初期化する
関数名	void **DMA_StructInit**(DMA_InitTypeDef* DMA_InitStruct)
機能	各 DMA_InitStruct メンバに初期値を書き込む
関数名	void **DMA_Cmd**(DMA_Channel_TypeDef* DMAy_Channelx, FunctionalState NewState)
機能	指定された DMA_Channelx を有効化/無効化する
関数名	void **DMA_ITConfig** (DMA_Channel_TypeDef* DMAy_Channelx, u32 DMA_IT, FunctionalState NewState)
機能	指定された DMA_Channelx 割り込みを有効化/無効化する
関数名	u16 **DMA_GetCurrDataCounter**(DMA_Channel_TypeDef* DMAy_Channelx)
機能	現在の DMA_Channelx 転送内に残っているデータ・ユニット数を返す
関数名	FlagStatus **DMA_GetFlagStatus**(u32 DMA_FLAG)
機能	指定された DMA_Channelx フラグがセットされたか否かを確認する
関数名	void **DMA_ClearFlag**(u32 DMA_FLAG)
機能	DMA_Channelx ペンディング・フラグをクリアする
関数名	ITStatus **DMA_GetITStatus**(u32 DMA_IT)
機能	指定された DMA チャネル x 割り込みが発生したか否かを確認する
関数名	void **DMA_ClearITPendingBit**(u32 DMA_IT)
機能	DMA チャネル割り込みペンディング・ビットをクリアする

フラッシュ・メモリ関連(stm32f10x_flash.h)

関数名	void **FLASH_SetLatency**(u32 FLASH_Latency)
機能	コード遅延値をセットする
関数名	void **FLASH_HalfCycleAccessCmd**(u32 FLASH_HalfCycleAccess)
機能	ハーフ・サイクル・フラッシュ・アクセスを有効化/無効化する
関数名	void **FLASH_PrefetchBufferCmd**(u32 FLASH_PrefetchBuffer)
機能	プリフェッチ・バッファを有効化/無効化する

汎用入出力関連(stm32f10x_gpio.h)

関数名	void **GPIO_DeInit**(GPIO_TypeDef* GPIOx)
機能	GPIO ペリフェラル・レジスタを初期値にリセットする
関数名	void **GPIO_AFIODeInit**(void)
機能	代替機能(リマップ，イベント・コントロール，EXTI 構成)レジスタを初期値にリセットする
関数名	void **GPIO_Init**(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct)
機能	GPIO_InitStruct 内にて指定されたパラメータに応じて，GPIO ペリフェラルを初期化する
関数名	void **GPIO_StructInit**(GPIO_InitTypeDef* GPIO_InitStruct)
機能	各 GPIO_InitStruct メンバに初期値を書き込む
関数名	u8 **GPIO_ReadInputDataBit**(GPIO_TypeDef* GPIOx, u16 GPIO_Pin)
機能	指定された入力ポート・ピンを読み出す

汎用入出力関連(stm32f10x_gpio.h)

関数名	u16 **GPIO_ReadInputData**(GPIO_TypeDef* GPIOx)
機能	指定された GPIO 入力データ・ポートを読み出す
関数名	u8 **GPIO_ReadOutputDataBit**(GPIO_TypeDef* GPIOx, u16 GPIO_Pin)
機能	指定された出力データ・ポート・ビットを読み出す
関数名	u16 **GPIO_ReadOutputData**(GPIO_TypeDef* GPIOx)
機能	指定された GPIO 出力データ・ポート・ビットを読み出す
関数名	void **GPIO_SetBits**(GPIO_TypeDef* GPIOx, u16 GPIO_Pin)
機能	選択されたデータ・ポート・ビットをセットする
関数名	void **GPIO_ResetBits**(GPIO_TypeDef* GPIOx, u16 GPIO_Pin)
機能	選択されたデータ・ポート・ビットをリセットする
関数名	void **GPIO_WriteBit**(GPIO_TypeDef* GPIOx, u16 GPIO_Pin, BitAction BitVal)
機能	選択されたデータ・ポート・ビットをセットまたはクリアする
関数名	void **GPIO_Write**(GPIO_TypeDef* GPIOx, u16 PortVal)
機能	指定されたデータを GPIO データ・ポートに書き込む
関数名	void **GPIO_PinLockConfig**(GPIO_TypeDef* GPIOx, u16 GPIO_Pin)
機能	GPIO ピン構成レジスタをロックする
関数名	void **GPIO_EventOutputConfig**(u8 GPIO_PortSource, u8 GPIO_PinSource)
機能	イベント出力として使用される GPIO ピンを選択する
関数名	void **GPIO_EventOutputCmd**(FunctionalState NewState)
機能	イベント出力を有効化/無効化する
関数名	void **GPIO_PinRemapConfig**(u32 GPIO_Remap, FunctionalState NewState)
機能	指定されたピンのマッピングを変更する
関数名	void **GPIO_EXTILineConfig**(u8 GPIO_PortSource, u8 GPIO_PinSource)
機能	EXTI ラインとして使用される GPIO ピンを選択する

多重割り込みベクタ・コントローラ関連(stm32f10x_nvic.h)

関数名	void **NVIC_DeInit**(void)
機能	NVIC ペリフェラル・レジスタを初期値にリセットする
関数名	void **NVIC_SCBDeInit**(void)
機能	SCB ペリフェラル・レジスタを初期値にリセットする
関数名	void **NVIC_PriorityGroupConfig**(uint32_t NVIC_PriorityGroup)
機能	優先順位グループを構成する:プリエンプションおよびサブ優先順位
関数名	void **NVIC_Init**(NVIC_InitTypeDef* NVIC_InitStruct)
機能	NVIC_InitStruct 内にてす指定されたパラメータに応じて，NVIC ペリフェラルを初期化する
関数名	void **NVIC_StructInit**(NVIC_InitTypeDef* NVIC_InitStruct)
機能	NVIC_InitStruct メンバに初期値を書き込む
関数名	void **NVIC_SETPRIMASK**(void)
機能	PRIMASK 優先順位を有効化する:実行優先順位を’0’に引き上げる
関数名	void **NVIC_RESETPRIMASK**(void)
機能	PRIMASK 優先順位を無効化する
関数名	void **NVIC_SETFAULTMASK**(void)
機能	FAULTMASK 優先順位を有効化する:実行優先順位を’1’引き下げる
関数名	void **NVIC_RESETFAULTMASK**(void)
機能	FAULTMASK 優先順位を無効化する
関数名	void **NVIC_BASEPRICONFIG**(u32 NewPriority)
機能	実行優先順位を’N’(構成可能な最低優先順位)から’1’に変更する
関数名	u32 **NVIC_GetBASEPRI**(void)
機能	BASEPRI マスク値を返す
関数名	u16 **NVIC_GetCurrentPendingIRQChannel**(void)
機能	現在中断している執行された IRQ チャネルの識別子を返す

多重割り込みベクタ・コントローラ関連(**stm32f10x_nvic.h**)

関数名	ITStatus **NVIC_GetIRQChannelPendingBitStatus**(u8 NVIC_IRQChannel)
機能	指定された IRQ チャネル・ペンディング・ビットがセットされたか否かを確認する
関数名	void **NVIC_SetIRQChannelPendingBit**(u8 NVIC_IRQChannel)
機能	NVIC 割り込みペンディング・ビットをセットする
関数名	void **NVIC_ClearIRQChannelPendingBit**(u8 NVIC_IRQChannel)
機能	NVIC 割り込みペンディング・ビットをクリアする
関数名	u16 **NVIC_GetCurrentActiveHandler**(void)
機能	現在のアクティブ・ハンドラ(IRQ チャネルおよびシステム・ハンドラ)識別子を返す
関数名	ITStatus **NVIC_GetIRQChannelActiveBitStatus**(u8 NVIC_IRQChannel)
機能	指定された IRQ チャネル・アクティブ・ビットがセットされたか否かを確認する
関数名	u32 **NVIC_GetCPUID**(void)
機能	ID 番号を返す : バージョン番号および Cortex-M3 コアの実装詳細
関数名	void **NVIC_SetVectorTable**(uint32_t NVIC_VectTab, uint32_t Offset)
機能	ベクタ・テーブル位置およびオフセットをセットする
関数名	void **NVIC_GenerateSystemReset**(void)
機能	システム・リセットを生成する
関数名	void **NVIC_GenerateCoreReset**(void)
機能	コア(コア＋NVIC)・リセットを生成する
関数名	void **NVIC_SystemLPConfig**(uint8_t LowPowerMode, FunctionalState NewState)
機能	低消費電力モードに入るためのシステムに対する条件を選択する
関数名	void **NVIC_SystemHandlerConfig**(u32 SystemHandler, FunctionalState NewState)
機能	指定されたシステム・ハンドラを有効化/無効化する
関数名	void **NVIC_SystemHandlerPriorityConfig** (u32 SystemHandler, u8 SystemHandlerPreemptionPriority, u8 SystemHandlerSubPriority)
機能	指定されたシステム・ハンドラ優先順位を構成する

多重割り込みベクタ・コントローラ関連(**stm32f10x_nvic.h**)

関数名	ITStatus **NVIC_GetSystemHandlerPendingBitStatus**(u32 SystemHandler)
機能	指定されたシステム・ハンドラ・ペンディング・ビットがセットされたか否かを確認する
関数名	void **NVIC_SetSystemHandlerPendingBit**(u32 SystemHandler)
機能	システム・ハンドラ・ペンディング・ビットをセットする
関数名	void **NVIC_ClearSystemHandlerPendingBit**(u32 SystemHandler)
機能	システム・ハンドラ・ペンディング・ビットをクリアする
関数名	ITStatus **NVIC_GetSystemHandlerActiveBitStatus**(u32 SystemHandler)
機能	指定されたシステム・ハンドラ・アクティブ・ビットがセットされたか否かを確認する
関数名	u32 **NVIC_GetFaultHandlerSources**(u32 SystemHandler)
機能	システム・フォルト・ハンドラ・ソースを返す
関数名	u32 **NVIC_GetFaultAddress**(u32 SystemHandler)
機能	フォルト・ハンドラを生成するアドレス位置を返す

電源コントロール関連(stm32f10x_pwr.h)

関数名	void **PWR_DeInit**(void)
機能	PWR ペリフェラル・レジスタを初期値へリセットする
関数名	void **PWR_BackupAccessCmd**(FunctionalState NewState)
機能	RTC およびバックアップ・レジスタへのアクセスを有効化/無効化する
関数名	void **PWR_PVDCmd**(FunctionalState NewState)
機能	電源電圧検出器を有効化/無効化する
関数名	void **PWR_PVDLevelConfig**(u32 PWR_PVDLevel)
機能	電源電圧検出器によって検出される電圧シュレッショルドを構成する
関数名	void **PWR_WakeUpPinCmd**(FunctionalState NewState)
機能	ウェイクアップ・ピン機能を有効化/無効化する
関数名	void **PWR_EnterSTOPMode**(u32 PWR_Regulator, u8 PWR_STOPEntry)
機能	STOP モードへ入る
関数名	void **PWR_EnterSTANDBYMode**(void)
機能	STANDBY モードへ入る
関数名	FlagStatus **PWR_GetFlagStatus**(u32 PWR_FLAG)
機能	指定された PWR フラグがセットされたか否かを確認する
関数名	void **PWR_ClearFlag**(u32 PWR_FLAG)
機能	PWR のペンディング・ビットをクリアする

リセットおよびクロック・コントローラ関連(stm32f10x_rcc.h)

関数名	void **RCC_DeInit**(void)
機能	RCC ペリフェラル・レジスタを初期値にリセットする
関数名	void **RCC_HSEConfig**(u32 RCC_HSE)
機能	外部高速オシレータ(HSE)を構成する
関数名	ErrorStatus **RCC_WaitForHSEStartUp**(void)
機能	HSE スタートアップを待つ
関数名	void **RCC_AdjustHSICalibrationValue**(u8 HSICalibrationValue)
機能	内蔵高速オシレータ(HSI)校正値を調整する
関数名	void **RCC_HSICmd**(FunctionalState NewState)
機能	内蔵高速オシレータ(HIS)を有効化/無効化する
関数名	void **RCC_PLLConfig**(u32 RCC_PLLSource, u32 RCC_PLLMul)
機能	PLL クロック・ソースおよび乗算因子を構成する
関数名	void **RCC_PLLCmd**(FunctionalState NewState)
機能	PLL を有効化/無効化する
関数名	void **RCC_SYSCLKConfig**(u32 RCC_SYSCLKSource)
機能	システム・クロック(SYSCLK)を構成する
関数名	u8 **RCC_GetSYSCLKSource**(void)
機能	システム・クロックとして使用されているクロック・ソースを返す
関数名	void **RCC_HCLKConfig**(u32 RCC_SYSCLK)
機能	AHB クロック(HCLK)を構成する
関数名	void **RCC_PCLK1Config**(u32 RCC_HCLK)
機能	低速 APB クロック(PCLK1)を構成する
関数名	void **RCC_PCLK2Config**(u32 RCC_HCLK)
機能	高速 APB クロック(PCLK2)を構成する

リセットおよびクロック・コントローラ関連(**stm32f10x_rcc.h**)

関数名	void **RCC_ITConfig**(u8 RCC_IT, FunctionalState NewState)
機能	指定された RCC 割り込みを有効化/無効化する
関数名	void **RCC_USBCLKConfig**(u32 RCC_USBCLKSource)
機能	USB クロック(USBCLK)を構成する
関数名	void **RCC_ADCCLKConfig**(u32 RCC_PCLK2)
機能	ADC クロック(ADCCLK)を構成する
関数名	void **RCC_LSEConfig**(u8 RCC_LSE)
機能	外部低速オシレータ(LSE)を構成する
関数名	void **RCC_LSICmd**(FunctionalState NewState)
機能	内蔵低速オシレータ(LSI)を有効化/無効化する
関数名	void **RCC_RTCCLKConfig**(u32 RCC_RTCCLKSource)
機能	RTC クロック(RTCCLK)を構成する
関数名	void **RCC_RTCCLKCmd**(FunctionalState NewState)
機能	RTC クロックを有効化/無効化する
関数名	void **RCC_GetClocksFreq**(RCC_ClocksTypeDef* RCC_Clocks)
機能	チップ・クロックにおける様々な周波数を返す
関数名	void **RCC_AHBPeriphClockCmd**(u32 RCC_AHBPeriph, FunctionalState NewState)
機能	AHB ペリフェラル・クロックを有効化/無効化する
関数名	void **RCC_APB2PeriphClockCmd**(u32 RCC_APB2Periph, FunctionalState NewState)
機能	高速 APB(APB2)ペリフェラル・クロックを有効化/無効化する
関数名	void **RCC_APB1PeriphClockCmd**(u32 RCC_APB1Periph, FunctionalState NewState)
機能	低速 APB(APB1)ペリフェラル・クロックを有効化/無効化する
関数名	void **RCC_APB2PeriphResetCmd**(u32 RCC_APB2Periph, FunctionalState NewState)
機能	高速 APB(APB2)ペリフェラル・リセットを強要もしくはリリースする

リセットおよびクロック・コントローラ関連(**stm32f10x_rcc.h**)

関数名	void **RCC_APB1PeriphResetCmd**(u32 RCC_APB1Periph, FunctionalState NewState)
機能	低速 APB(APB1)ペリフェラル・リセットを強要もしくはリリースする
関数名	void **RCC_BackupResetCmd**(FunctionalState NewState)
機能	バックアップ領域リセットを強要もしくはリリースする
関数名	void **RCC_ClockSecuritySystemCmd**(FunctionalState NewState)
機能	クロック・セキュリティ・システムを有効化/無効化する
関数名	void **RCC_MCOConfig**(u8 RCC_MCO)
機能	MCO ピン上に出力するために，クロックソースを選択する
関数名	FlagStatus **RCC_GetFlagStatus**(u8 RCC_FLAG)
機能	指定された RCC フラグがセットされたか否かを確認する
関数名	void **RCC_ClearFlag**(void)
機能	RCC リセット・フラグをクリアする
関数名	ITStatus **RCC_GetITStatus**(u8 RCC_IT)
機能	指定された RCC 割り込みが発生したか否かを確認する
関数名	void **RCC_ClearITPendingBit**(u8 RCC_IT)
機能	RCC の割り込みペンディング・ビットをクリアする

SysTick 関連(stm32f10x_systick.h)

関数名	void **SysTick_CLKSourceConfig**(uint32_t SysTick_CLKSource)
機能	SysTick のクロック・ソースを構成する
関数名	void **SysTick_SetReload**(u32 Reload)
機能	SysTick 再ロード値をセットする
関数名	void **SysTick_CounterCmd**(u32 SysTick_Counter)
機能	SysTick カウンタを有効化/無効化する
関数名	void **SysTick_ITConfig**(FunctionalState NewState)
機能	SysTick 割り込みを有効化/無効化する
関数名	u32 **SysTick_GetCounter**(void)
機能	SysTick カウンタ値を取得する
関数名	FlagStatus **SysTick_GetFlagStatus**(u8 SysTick_FLAG)
機能	指定された SysTick フラグがセットされたか否かを確認する

汎用タイマ関連(stm32f10x_tim.h)

関数名	void **TIM_DeInit**(TIM_TypeDef* TIMx)
機能	TIMx ペリフェラル・レジスタを初期値にリセットする
関数名	void **TIM_TimeBaseInit** (TIM_TypeDef* TIMx, TIM_TimeBaseInitTypeDef* TIM_TimeBaseInitStruct)
機能	TIM_TimeBaseInit Struct 内にて指定されたパラメータに応じて，TIMxタイム・ベース・ユニットを初期化する
関数名	void **TIM_OC1Init**(TIM_TypeDef* TIMx, TIM_OCInitTypeDef* TIM_OCInitStruct)
機能	TIM_OCInitStruct 内にて指定されたパラメータに応じて，TIMxペリフェラルを初期化する
関数名	void **TIM_OC2Init**(TIM_TypeDef* TIMx, TIM_OCInitTypeDef* TIM_OCInitStruct)
機能	TIM_OCInitStruct 内にて指定されたパラメータに応じて，TIMxペリフェラルを初期化する
関数名	void **TIM_OC3Init**(TIM_TypeDef* TIMx, TIM_OCInitTypeDef* TIM_OCInitStruct)
機能	TIM_OCInitStruct 内にて指定されたパラメータに応じて，TIMxペリフェラルを初期化する
関数名	void **TIM_OC4Init**(TIM_TypeDef* TIMx, TIM_OCInitTypeDef* TIM_OCInitStruct)
機能	TIM_OCInitStruct 内にて指定されたパラメータに応じて，TIMxペリフェラルを初期化する
関数名	void **TIM_ICInit**(TIM_TypeDef* TIMx, TIM_ICInitTypeDef* TIM_ICInitStruct)
機能	TIM_ICInitStruct 内にて指定されたパラメータに応じて，TIMxペリフェラルを初期化する
関数名	void **TIM_PWMIConfig**(TIM_TypeDef* TIMx, TIM_ICInitTypeDef* TIM_ICInitStruct)
機能	TIM_OCInitStruct 内にて指定されたパラメータに応じて，PWM 入力モードにおける TIMxペリフェラルを初期化する
関数名	void **TIM_BDTRConfig** (TIM_TypeDef* TIMx, TIM_BDTRInitTypeDef *TIM_BDTRInitStruct)
機能	ブレーク機能・デッドタイム・ロック・レベル，OSSI，OSSR ステータスおよび AOE(Automatic Output Enable)を構成する
関数名	void **TIM_TimeBaseStructInit**(TIM_TimeBaseInitTypeDef* TIM_TimeBaseInitStruct)
機能	各 TIM_TimeBaseInitStruct メンバに初期値を書き込む
関数名	void **TIM_OCStructInit**(TIM_OCInitTypeDef* TIM_OCInitStruct)
機能	各 TIM_OCInitStruct メンバに初期値を書き込む
関数名	void **TIM_ICStructInit**(TIM_ICInitTypeDef* TIM_ICInitStruct)
機能	各 TIM_ICInitStruct メンバに初期値を書き込む

汎用タイマ関連(**stm32f10x_tim.h**)

関数名	void **TIM_BDTRStructInit**(TIM_BDTRInitTypeDef* TIM_BDTRInitStruct)
機能	各 TIM_BDTRInitStruct メンバに初期値を書き込む
関数名	void **TIM_Cmd**(TIM_TypeDef* TIMx, FunctionalState NewState)
機能	指定された TIM ペリフェラルを有効化/無効化する
関数名	void **TIM_CtrlPWMOutputs**(TIM_TypeDef* TIMx, FunctionalState NewState)
機能	TIMxペリフェラル・メイン出力を有効化/無効化する
関数名	void **TIM_ITConfig**(TIM_TypeDef* TIMx, u16 TIM_IT, FunctionalState NewState)
機能	指定された TIM 割り込みを有効化/無効化する
関数名	void **TIM_GenerateEvent**(TIM_TypeDef* TIMx, u16 TIM_EventSource)
機能	ソフトウェアによって生成される TIMxイベントを構成する
関数名	void **TIM_DMAConfig** (TIM_TypeDef* TIMx, u16 TIM_DMABase, u16 TIM_DMABurstLength)
機能	TIMx_DMA インターフェースを構成する
関数名	void **TIM_DMACmd**(TIM_TypeDef* TIMx, u16 TIM_DMASource, FunctionalState NewState)
機能	TIMx_DMA 要求を有効化/無効化する
関数名	void **TIM_InternalClockConfig**(TIM_TypeDef* TIMx)
機能	TIMx内部クロックを構成する
関数名	void **TIM_ITRxExternalClockConfig**(TIM_TypeDef* TIMx, u16 TIM_InputTriggerSource)
機能	TIMx内部トリガを外部クロックとして構成する
関数名	void **TIM_TIxExternalClockConfig** (TIM_TypeDef* TIMx, u16 TIM_TIxExternalCLKSource,u16 TIM_ICPolarity, u16 ICFilter)
機能	TIMxトリガを外部クロックとして構成する
関数名	void **TIM_ETRClockMode1Config** (TIM_TypeDef* TIMx, u16 TIM_ExtTRGPrescaler, u16 TIM_ExtTRGPolarity,u16ExtTRGFilter)
機能	TIMx外部クロック・モード 1 を構成する
関数名	void **TIM_ETRClockMode2Config** (TIM_TypeDef* TIMx, u16 TIM_ExtTRGPrescaler, u16 TIM_ExtTRGPolarity, u16 ExtTRGFilter)
機能	TIMx外部クロック・モード 2 を構成する

汎用タイマ関連(stm32f10x_tim.h)

関数名	void **TIM_ETRConfig** (TIM_TypeDef* TIMx, u16 TIM_ExtTRGPrescaler, u16 TIM_ExtTRGPolarity,u16 ExtTRGFilter)
機能	TIMx外部トリガ(ETR)を構成する
関数名	void **TIM_PrescalerConfig** (TIM_TypeDef* TIMx, u16 Prescaler, u16 TIM_PSCReloadMode)
機能	TIMxプリスケーラを構成する
関数名	void **TIM_CounterModeConfig**(TIM_TypeDef* TIMx, u16 TIM_CounterMode)
機能	使用される TIMxカウンタ・モードを指定する
関数名	void **TIM_SelectInputTrigger**(TIM_TypeDef* TIMx, u16 TIM_InputTriggerSource)
機能	TIMx入力トリガ・ソースを選択する
関数名	void **TIM_EncoderInterfaceConfig** (TIM_TypeDef* TIMx, u16 TIM_EncoderMode,u16 TIM_IC1Polarity, u16 TIM_IC2Polarity)
機能	TIMxエンコーダ・インターフェースを構成する
関数名	void **TIM_ForcedOC1Config**(TIM_TypeDef* TIMx, u16 TIM_ForcedAction)
機能	TIMx出力 1 波形をアクティブ/インアクティブ・レベルに強要する
関数名	void **TIM_ForcedOC2Config**(TIM_TypeDef* TIMx, u16 TIM_ForcedAction)
機能	TIMx出力 2 波形をアクティブ/インアクティブ・レベルに強要する
関数名	void **TIM_ForcedOC3Config**(TIM_TypeDef* TIMx, u16 TIM_ForcedAction)
機能	TIMx出力 3 波形をアクティブ/インアクティブ・レベルに強要する
関数名	void **TIM_ForcedOC4Config**(TIM_TypeDef* TIMx, u16 TIM_ForcedAction)
機能	TIMx出力 4 波形をアクティブ/インアクティブ・レベルに強要する
関数名	void **TIM_ARRPreloadConfig**(TIM_TypeDef* TIMx, FunctionalState NewState)
機能	ARR 上の TIMxペリフェラル再ロード・レジスタを有効化/無効化する
関数名	void **TIM_SelectCOM**(TIM_TypeDef* TIMx, FunctionalState NewState)
機能	TIMxペリフェラル通信イベントを選択する
関数名	void **TIM_SelectCCDMA**(TIM_TypeDef* TIMx, FunctionalState NewState)
機能	TIMxペリフェラル・キャプチャ・コンペア DMA ソースを選択する

汎用タイマ関連(stm32f10x_tim.h)

関数名	void **TIM_CCPreloadControl**(TIM_TypeDef* TIMx, FunctionalState NewState)
機能	TIMxペリフェラル・キャプチャ・コンペア・再ロード・コントロール・ビットをセット/リセットする
関数名	void **TIM_OC1PreloadConfig**(TIM_TypeDef* TIMx, u16 TIM_OCPreload)
機能	CCR1 上の TIMxペリフェラル再ロード・レジスタを有効化/無効化する
関数名	void **TIM_OC2PreloadConfig**(TIM_TypeDef* TIMx, u16 TIM_OCPreload)
機能	CCR2 上の TIMxペリフェラル再ロード・レジスタを有効化/無効化する
関数名	void **TIM_OC3PreloadConfig**(TIM_TypeDef* TIMx, u16 TIM_OCPreload)
機能	CCR3 上の TIMxペリフェラル再ロード・レジスタを有効化/無効化する
関数名	void **TIM_OC4PreloadConfig**(TIM_TypeDef* TIMx, u16 TIM_OCPreload)
機能	CCR4 上の TIMxペリフェラル再ロード・レジスタを有効化/無効化する
関数名	void **TIM_OC1FastConfig**(TIM_TypeDef* TIMx, u16 TIM_OCFast)
機能	TIMx出力コンペア 1 高速機能を構成する
関数名	void **TIM_OC2FastConfig**(TIM_TypeDef* TIMx, u16 TIM_OCFast)
機能	TIMx出力コンペア 2 高速機能を構成する
関数名	void **TIM_OC3FastConfig**(TIM_TypeDef* TIMx, u16 TIM_OCFast)
機能	TIMx出力コンペア 3 高速機能を構成する
関数名	void **TIM_OC4FastConfig**(TIM_TypeDef* TIMx, u16 TIM_OCFast)
機能	TIMx出力コンペア 4 高速機能を構成する
関数名	void **TIM_ClearOC1Ref**(TIM_TypeDef* TIMx, u16 TIM_OCClear)
機能	外部イベント上の OCREF1 信号をクリアもしくは保護する
関数名	void **TIM_ClearOC2Ref**(TIM_TypeDef* TIMx, u16 TIM_OCClear)
機能	外部イベント上の OCREF2 信号をクリアもしくは保護する
関数名	void **TIM_ClearOC3Ref**(TIM_TypeDef* TIMx, u16 TIM_OCClear)
機能	外部イベント上の OCREF3 信号をクリアもしくは保護する
関数名	void **TIM_ClearOC4Ref**(TIM_TypeDef* TIMx, u16 TIM_OCClear)
機能	外部イベント上の OCREF4 信号をクリアもしくは保護する

汎用タイマ関連(stm32f10x_tim.h)

関数名	void **TIM_OC1PolarityConfig**(TIM_TypeDef* TIMx, u16 TIM_OCPolarity)
機能	TIMxチャネル 1 の極性を構成する
関数名	void **TIM_OC1NPolarityConfig**(TIM_TypeDef* TIMx, u16 TIM_OCNPolarity)
機能	TIMxチャネル 1 の N 極性を構成する
関数名	void **TIM_OC2PolarityConfig**(TIM_TypeDef* TIMx, u16 TIM_OCPolarity)
機能	TIMxチャネル 2 の極性を構成する
関数名	void **TIM_OC2NPolarityConfig**(TIM_TypeDef* TIMx, u16 TIM_OCNPolarity)
機能	TIMxチャネル 2 の N 極性を構成する
関数名	void **TIM_OC3PolarityConfig**(TIM_TypeDef* TIMx, u16 TIM_OCPolarity)
機能	TIMxチャネル 3 の極性を構成する
関数名	void **TIM_OC3NPolarityConfig**(TIM_TypeDef* TIMx, u16 TIM_OCNPolarity)
機能	TIMxチャネル 3 の N 極性を構成する
関数名	void **TIM_OC4PolarityConfig**(TIM_TypeDef* TIMx, u16 TIM_OCPolarity)
機能	TIMxチャネル 4 の極性を構成する
関数名	void **TIM_CCxCmd**(TIM_TypeDef* TIMx, u16 TIM_Channel, u16 TIM_CCx)
機能	TIMxキャプチャ・コンペア・チャネルxを有効化/無効化する
関数名	void **TIM_CCxNCmd**(TIM_TypeDef* TIMx, u16 TIM_Channel, u16 TIM_CCxN)
機能	TIMxキャプチャ・コンペア・チャネルxN を有効化/無効化する
関数名	void **TIM_SelectOCxM**(TIM_TypeDef* TIMx, u16 TIM_Channel, u16 TIM_OCMode)
機能	TIMx出力コンペア・モードを選択する
関数名	void **TIM_UpdateDisableConfig**(TIM_TypeDef* TIMx, FunctionalState NewState)
機能	TIMx更新イベントを有効化/無効化する
関数名	void **TIM_UpdateRequestConfig**(TIM_TypeDef* TIMx, u16 TIM_UpdateSource)
機能	TIMx更新要求ソースを構成する
関数名	void **TIM_SelectHallSensor**(TIM_TypeDef* TIMx, FunctionalState NewState)
機能	TIMxホール・センサ・インターフェースを有効化/無効化する
関数名	void **TIM_SelectOnePulseMode**(TIM_TypeDef* TIMx, u16 TIM_OPMode)
機能	TIMxワン・パルス・モードを選択する

汎用タイマ関連(stm32f10x_tim.h)

関数名	void **TIM_SelectOutputTrigger**(TIM_TypeDef* TIMx, u16 TIM_TRGOSource)
機能	TIMxトリガ出力モードを選択する
関数名	void **TIM_SelectSlaveMode**(TIM_TypeDef* TIMx, u16 TIM_SlaveMode)
機能	TIMxスレーブ・モードを選択する
関数名	void **TIM_SelectMasterSlaveMode**(TIM_TypeDef* TIMx, u16 TIM_MasterSlaveMode)
機能	TIMxマスタ/スレーブ・モードをセット/リセットする
関数名	void **TIM_SetCounter**(TIM_TypeDef* TIMx, u16 Counter)
機能	TIMxカウンタ・レジスタ値をセットする
関数名	void **TIM_SetAutoreload**(TIM_TypeDef* TIMx, u16 Autoreload)
機能	TIMx自動再ロード・レジスタ値をセットする
関数名	void **TIM_SetCompare1**(TIM_TypeDef* TIMx, u16 Compare1)
機能	TIMxキャプチャ・コンペア 1 レジスタ値をセットする
関数名	void **TIM_SetCompare2**(TIM_TypeDef* TIMx, u16 Compare2)
機能	TIMxキャプチャ・コンペア 2 レジスタ値をセットする
関数名	void **TIM_SetCompare3**(TIM_TypeDef* TIMx, u16 Compare3)
機能	TIMxキャプチャ・コンペア 3 レジスタ値をセットする
関数名	void **TIM_SetCompare4**(TIM_TypeDef* TIMx, u16 Compare4)
機能	TIMxキャプチャ・コンペア 4 レジスタ値をセットする
関数名	void **TIM_SetIC1Prescaler**(TIM_TypeDef* TIMx, u16 TIM_ICPSC)
機能	TIMx入力キャプチャ 1 プリスケーラをセットする
関数名	void **TIM_SetIC2Prescaler**(TIM_TypeDef* TIMx, u16 TIM_ICPSC)
機能	TIMx入力キャプチャ 2 プリスケーラをセットする
関数名	void **TIM_SetIC3Prescaler**(TIM_TypeDef* TIMx, u16 TIM_ICPSC)
機能	TIMx入力キャプチャ 3 プリスケーラをセットする
関数名	void **TIM_SetIC4Prescaler**(TIM_TypeDef* TIMx, u16 TIM_ICPSC)
機能	TIMx入力キャプチャ 4 プリスケーラをセットする

汎用タイマ関連(stm32f10x_tim.h)

関数名	void **TIM_SetClockDivision**(TIM_TypeDef* TIMx, u16 TIM_CKD)
機能	TIMxクロック分周期をセットする
関数名	u16 **TIM_GetCapture1**(TIM_TypeDef* TIMx)
機能	TIMx入力キャプチャ 1 値を取得する
関数名	u16 **TIM_GetCapture2**(TIM_TypeDef* TIMx)
機能	TIMx入力キャプチャ 2 値を取得する
関数名	u16 **TIM_GetCapture3**(TIM_TypeDef* TIMx)
機能	TIMx入力キャプチャ 3 値を取得する
関数名	u16 **TIM_GetCapture4**(TIM_TypeDef* TIMx)
機能	TIMx入力キャプチャ 4 値を取得する
関数名	u16 **TIM_GetCounter**(TIM_TypeDef* TIMx)
機能	TIMxカウンタ値を取得する
関数名	u16 **TIM_GetPrescaler**(TIM_TypeDef* TIMx)
機能	TIMxプリスケーラ値を取得する
関数名	FlagStatus **TIM_GetFlagStatus**(TIM_TypeDef* TIMx, u16 TIM_FLAG)
機能	指定された TIM 割り込みが発生したか否か確認する
関数名	void **TIM_ClearFlag**(TIM_TypeDef* TIMx, u16 TIM_FLAG)
機能	TIMxペンディング・フラグをクリアする
関数名	ITStatus **TIM_GetITStatus**(TIM_TypeDef* TIMx, u16 TIM_IT)
機能	指定された TIMxフラグがセットされたか否かを確認する
関数名	void **TIM_ClearITPendingBit**(TIM_TypeDef* TIMx, u16 TIM_IT)
機能	TIMx割り込みペンディング・ビットをクリアする

USART 関連(stm32f10x_usart.h)

関数名	void **USART_DeInit**(USART_TypeDef* USARTx)
機能	USARTx ペリフェラルを初期値にリセットする
関数名	void **USART_Init**(USART_TypeDef* USARTx, USART_InitTypeDef* USART_InitStruct)
機能	USART_InitStruct 内にて指定されたパラメータに応じて，USARTx ペリフェラルを初期化する
関数名	void **USART_StructInit**(USART_InitTypeDef* USART_InitStruct)
機能	各 USART_InitStruct メンバに初期値を書き込む
関数名	void **USART_ClockInit** (USART_TypeDef* USARTx, USART_ClockInitTypeDef* USART_ClockInitStruct)
機能	USARTx クロック・ペリフェラルを初期値にリセットする
関数名	void **USART_ClockStructInit**(USART_ClockInitTypeDef* USART_ClockInitStruct)
機能	USART_ClockInitStructure 内にて指定されたパラメータに応じて，USARTxクロック・ペリフェラルを初期化する
関数名	void **USART_Cmd**(USART_TypeDef* USARTx, FunctionalState NewState)
機能	指定された USART ペリフェラルを有効化/無効化する
関数名	void **USART_ITConfig** (USART_TypeDef* USARTx, u16 USART_IT, FunctionalState NewState)
機能	指定された USART 割り込みを有効化/無効化する
関数名	void **USART_DMACmd** (USART_TypeDef* USARTx, u16 USART_DMAReq, FunctionalState NewState)
機能	USART_DMA インターフェースを有効化/無効化する
関数名	void **USART_SetAddress**(USART_TypeDef* USARTx, u8 USART_Address)
機能	USART ノードのアドレスをセットする
関数名	void **USART_WakeUpConfig**(USART_TypeDef* USARTx, u16 USART_WakeUp)
機能	USART ウェイクアップ方法を選択する
関数名	void **USART_ReceiverWakeUpCmd** (USART_TypeDef* USARTx, FunctionalState NewState)
機能	USART がミュート・モードか否かを確認する
関数名	void **USART_LINBreakDetectLengthConfig** (USART_TypeDef* USARTx, u16 USART_LINBreakDetectLength)
機能	USART_LIN ブレーク検出長をセットする
関数名	void **USART_LINCmd**(USART_TypeDef* USARTx, FunctionalState NewState)
機能	USARTx_LIN モードを有効化/無効化する

USART 関連(stm32f10x_usart.h)

関数名	void **USART_SendData**(USART_TypeDef* USARTx, u16 Data)
機能	USARTxペリフェラルにより，単一データを送信する
関数名	u16 **USART_ReceiveData**(USART_TypeDef* USARTx)
機能	USARTxペリフェラルによる直近の受信データを返す
関数名	void **USART_SendBreak**(USART_TypeDef* USARTx)
機能	ブレーク・キャラクタを送信する
関数名	void **USART_SetGuardTime**(USART_TypeDef* USARTx, u8 USART_GuardTime)
機能	指定された USART ガード・タイムをセットする
関数名	void **USART_SetPrescaler**(USART_TypeDef* USARTx, u8 USART_Prescaler)
機能	USART クロック・プリスケーラをセットする
関数名	void **USART_SmartCardCmd**(USART_TypeDef* USARTx, FunctionalState NewState)
機能	USART スマート・カード・モードを有効化/無効化する
関数名	void **USART_SmartCardNACKCmd**(USART_TypeDef* USARTx, FunctionalState NewState)
機能	NACK 送信を有効化/無効化する
関数名	void **USART_HalfDuplexCmd**(USART_TypeDef* USARTx, FunctionalState NewState)
機能	USART 半二重モードを有効化/無効化する
関数名	void **USART_IrDAConfig**(USART_TypeDef* USARTx, u16 USART_IrDAMode)
機能	USART_IrDA モードを構成する
関数名	void **USART_IrDACmd**(USART_TypeDef* USARTx, FunctionalState NewState)
機能	USART_IrDA モードを有効化/無効化する
関数名	FlagStatus **USART_GetFlagStatus**(USART_TypeDef* USARTx, u16 USART_FLAG)
機能	指定された USART フラグがセットされたか否かを確認する
関数名	void **USART_ClearFlag**(USART_TypeDef* USARTx, u16 USART_FLAG)
機能	USARTxペンディング・フラグをクリアする
関数名	ITStatus **USART_GetITStatus**(USART_TypeDef* USARTx, u16 USART_IT)
機能	指定された USART 割り込みが発生したか否かを確認する
関数名	void **USART_ClearITPendingBit**(USART_TypeDef* USARTx, u16 USART_IT)
機能	USARTx割り込みペンディング・ビットをクリアする

3. USB ライブラリ

USB ライブラリ・コア・モジュール

ファイル名	**usb_type.h**
説明	ライブラリコアで使用されるタイプ 本ファイルは USB ライブラリの非依存性を保証する為に使用される
ファイル名	**usb_regs.h, usb_regs.c**
説明	ハードウェア・アブストラクション層
ファイル名	**usb_int.h, usb_int.c**
説明	転送割り込みサービス・ルーチン
ファイル名	**usb_init.h　usb_init.c**
説明	USB の初期化
ファイル名	**usb_core.h　usb_core.c**
説明	USB プロトコルの管理
ファイル名	**usb_mem.h　usb_mem.c**
説明	データ転送の管理
ファイル名	**usb_def.h**
説明	USB の初期化定義

アプリケーション・インターフェース・モジュール

ファイル名	**usb_istr.h　　usb_istr.c**
説明	USB 割り込みハンドら関数
ファイル名	**usb_conf.h**
説明	USB 構成ファイル
ファイル名	**usb_prop.h　　usb_prop.c**
説明	USB アプリケーション特有のプロパティ
ファイル名	**usb_endp.c**
説明	ノン・コントロール・エンドポイント用 CTR 割り込みハンドラ・ルーチン
ファイル名	**usb_pwr.h　　usb_pwr.c**
説明	USB 電源管理モジュール
ファイル名	**usb_desc.h　　usb_desc.c**
説明	USB ディスクリプション

その他

＜商標について＞

本ドキュメントに掲載されている会社名,製品名は，それぞれ各社の商標または登録商標です.

＜免責事項＞

- 本資料は，参考資料として公開されるものです．本資料中に記載の技術情報について弊社または第三者の知的財産権その他の権利の実施，使用を許諾または保証するものではありません.
- 本資料に記載の製品データ，図，表，プログラム，アルゴリズムその他応用回路例など全ての情報の使用に起因する損害，第三者の知的財産権その他の権利に対する侵害に関し弊社は責任を負いません.
- 本資料に記載の製品データ，図，表，プログラム，アルゴリズムその他応用回路例などの全ての情報は本資料発行時点のものであり，弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります.
- 本資料に記載した情報は，正確を期すため慎重に制作したものですが，万一本資料の記述の誤りに起因する損害がお客様に生じた場合においても，弊社はその責任を負いません.
- 本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断り致します.

TECHNO ROAD Inc.
株式会社テクノロード
〒213-0012
神奈川県川崎市高津区坂戸 3-2-1 KSP 西棟 4 階 NEO G-1
http://techno-road.com/ E-mail:post@techno-road.com