TOPPERS 活用アイデア・アプリケーション開発コンテスト

部門　：　活用アイデア部門
　　アプリケーション開発部門

作品のタイトル　：　ワンチップマイコンによる画像処理システム

作成者　：　羽佐田 理恵　（個人）

共同作業者　：　ＦＣＴ）組込みシステム研究サークル

対象者　：　画像処理プログラムに興味のある人、画像処理の初学者

使用する開発成果物　：　TOPPERS/ASP カーネル

目的・狙い

画像処理システムというと WindowsPC や LinuxPC など高機能、高性能なコンピュータを用いることが多いですが、こういったシステムでは既存のアプリやライブラリに依存したプログラミングになりやすく、基本部分の理解がおろそかになりがちです。データの扱いや画像処理方法が理解しやすいよう組込み向けワンチップマイコンによる画像処理システムを開発し、データの取り込みから画像処理、生成した画像の表示や保存までを自身で理解しながら行える環境を作ろうと考えました。

アイデア/アプリケーションの概要

STM32F746G-DISCO ボードにカメラ、２つのグラフィック LCD、操作用ボタンを追加し、LCD への表示や SD カードのリード／ライトを API としてユーザに提供しています。動作はカメラから画像を取り込みリアルタイムでオンボード LCD へ表示しながら操作用ボタンでユーザ関数を呼び出します。ユーザは API を使い画像処理結果の表示や保存をすることができます。また、画像処理の試行としてカメラで２度撮影した差分と SD カード上の背景用 BMP ファイルを合成して結果をグラフィック LCD へ表示するとともに SD カードに BMP ファイルとして保存するプログラムを作りました。

１．開発の背景

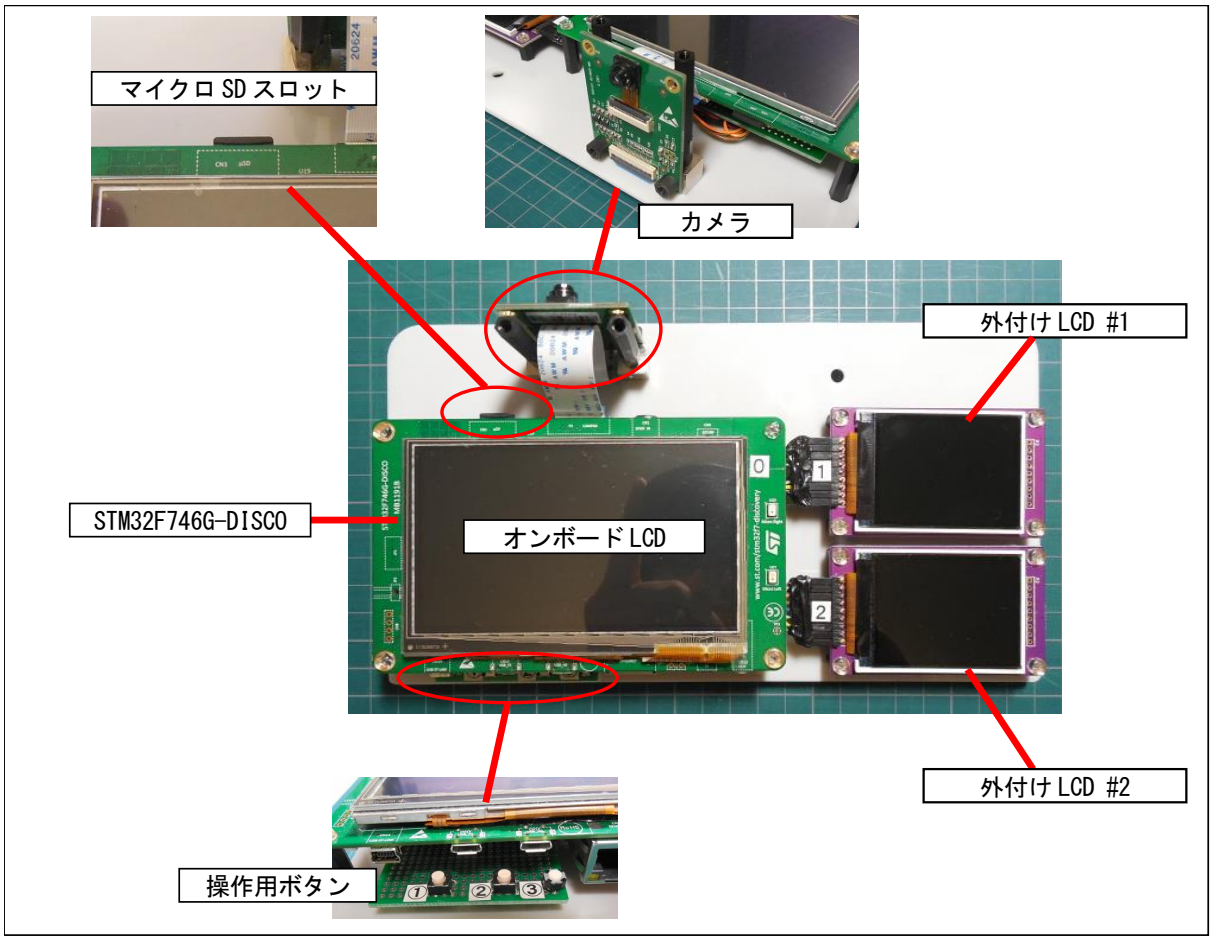
業務として Windows PC 上で動作する画像や映像を処理するプログラムの開発に携わっていますが、ハードウェア面では高性能な CPU や大容量のメモリ、ディスクへのファイルのリードライト、大画面への表示とリッチな環境で、ソフトウェアも画像処理を補助する豊富なライブラリが使えるなど、プログラム開発者としては便利な環境がそろっています。

一方、近年ではエッジコンピューティングなどとして末端の機器で画像や映像などを簡易に処理するシステムの要望が高まっています。エッジコンピューティングでは CPU の能力やメモリ、ディスク、画面表示などで制約が多くなり、利用できるライブラリも限定されます。このため、画像処理の基本をおさえながらプログラムを開発する技術が重要になると考えました。

また、社内の他部署では組込み向けの開発環境や機器の開発を行っていて、組込み用途のソフトウェアの開発技術も身に付けていく必要があります。

これらの背景に加え、会社の先輩が STM32F746G-DISC とカメラを用いた装置で２０１７年の TOPPERS コンテストで入賞しています。STM32F746G-DISC はワンチップマイコンを搭載したボードでありながら強力なマイコンや大容量の RAM、グラフィック LCD、マイクロ SD カードスロットなど画像処理に有用なデバイスを各種搭載しています。TOPPERS/ASP カーネルの動作実績や開発環境などの入手も容易であることから STM32F746G-DISC を用いた画像処理システムを開発することとしました。

２．開発システムの紹介



●STM32F746G-DISCO

ST マイクロエレクトロニクス社製の Cortex-M7 の STM32F746 が搭載されたマイコンボード。
後で説明するオンボード LCD、マイクロ SD スロットのほか、外付けの Flash メモリ、SDRAM、カメラインターフェース、タッチパネル、MEMS マイクなどが実装。
http://akizukidenshi.com/catalog/g/gM-09880/

●オンボード LCD

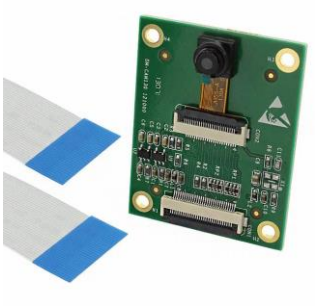
4.3 インチ、480x272 ドットのグラフィックカラーLCD。 STM32F746 の LCDC で制御。

●マイクロ SD スロット

マイクロ SD カードが差せるスロット。STM32F746 の SDMMC で SD カードを制御。

●カメラ

OV9655 を搭載したカメラ。最大 1280x1024 ピクセルの画像が取得できるが、当装置では 320x240 ピクセルの QVGA サイズでの画像取得を行う。STM32F746 の DCMI でカメラを制御。
https://www.digikey.jp/product-detail/ja/stmicroelectronics/STM32F4DIS-CAM/497-13546-ND/3878237



●外付け LCD #1、#2

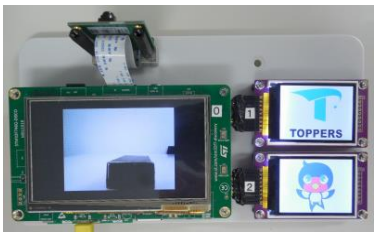
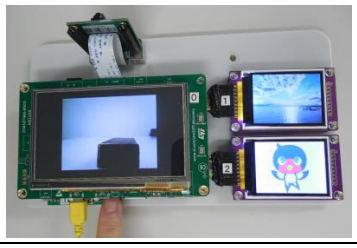
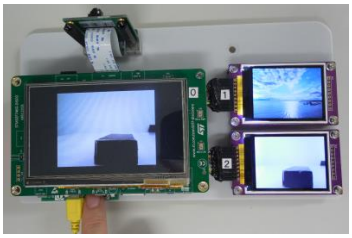
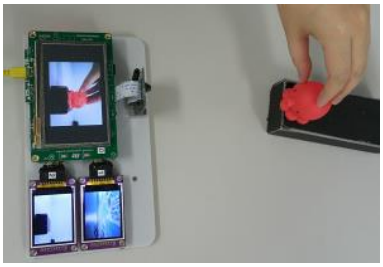
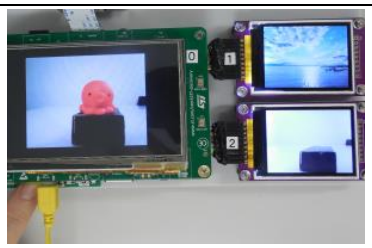
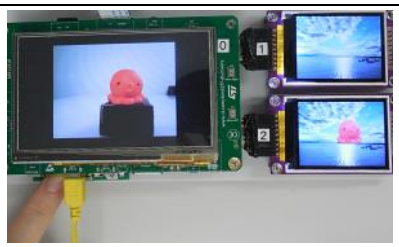
コントローラに ILI9328 を用いた 320x240 ドット、SPI インタフェースのグラフィックカラーLCD。
STM32F746 の SPI インタフェース、および、CS、RST は GPIO に接続して制御。
http://www.aitendo.com/product/10943
オンボード LCD だけでは画像処理結果の表示などで不便なため 2 つの LCD を外付けで追加。

●操作用ボタン

プッシュスイッチ 3 つを用いて今開発で作成したボタン。GPIO に接続して ON/OFF 状態を取得。
ユーザ作成関数の呼び出しに用いる。

● 試行プログラムの紹介

開発した画像処理システムの動作確認として、「被写体なし」と「被写体あり」の２枚の画像を撮影し、差分から被写体を抽出して事前にSDカード内に用意したBMPファイルの背景画像と重ね合わせて表示し、またSDカードへBMPファイルへ保存するプログラムを作成しました。

（１）電源投入 電源はボードのUSBコネクタから供給します。電源を投入するとオンボードLCDにはカメラで撮影した画像がリアルタイムで表示されます。 外付けLCDへは動作確認を兼ねて「TOPPERSロゴ」と「とばめ」の画像が表示されます。		
（２）背景画像選択 操作ボタン#3を押して重ね合わせた時の背景画像を選択します。画像は外付けLCD#1へ表示されます。 背景画像は24ビットカラーの320x240ピクセルのBMPファイルが利用できます。ファイル名は bg320x240_1.bmp ～ bg320x240_10.bmp にすると利用することができます。		
（３）「被写体なし」の画像撮影 被写体のない状態で操作ボタン#2を押すと「被写体なし」画像を撮影することができます。「被写体あり」画像を撮影するまで、何度撮影しても構いません。最後に撮影した画像が「被写体なし」画像として使われます。		
（４）「被写体」の配置 オンボードLCDを見ながら被写体を配置します。 このときボードを動かしてしまった場合には、前項（３）から操作をやり直してください。		
（５）「被写体あり」の画像撮影 被写体の構図が決まったら操作ボタン#1を押すことで「被写体あり」画像を撮影することができます。		
（６）重ね合わせ画像出力 重ね合わせた画像が外付けLCD#2へ出力されます。またSDカードへもPnnnnnn.bmp（nは６桁の数字、例：P000011.bmp）で保存されます。 この項の右側の画像が保存された画像です。		

動作動画も撮影し、以下のURLから閲覧できるようにしました。

https://youtu.be/ImkB7MSyY9c

３．開発環境

開発環境は ubuntu 16.04-32bit 版が動作する PC に ARM のクロスコンパイラをインストールし、TOPPERS/ASP ターゲット非依存部と ARM Cortex-M7 アーキテクチャ・GCC 依存パッケージを展開して使用しました。

展開した TOPPERS/ASP カーネルを用いてアプリケーションやデバイスアクセス処理を作成し、ARM のクロスコンパイラでビルドすることで実行プログラムを生成しています。

デバイスアクセス処理の一部はＳＴマイクロエレクトロニクスが提供する STM32CubeF7 パッケージからデバイス用ソースコードと FatFs を流用しています。

使用したソフトウェアなどは以下の通りです。

●TOPPERS/ASP カーネル　ターゲット非依存部
TOPPERS/ASP カーネルの共通部分のソース
< https://www.toppers.jp/download.cgi/asp-1.9.3.tar.gz >

●TOPPERS/ASP カーネル　ターゲット依存部
ARM Cortex-M7 アーキテクチャ・GCC 依存部パッケージ
< https://www.toppers.jp/download.cgi/asp_arch_arm_m7_gcc-1.9.3.tar.gz >

●STM32CubeF7
ST マイクロエレクトロニクスが提供するサンプルコードのパッケージ
< https://www.st.com/ja/embedded-software/stm32cubef7.html >
※含む FatFs

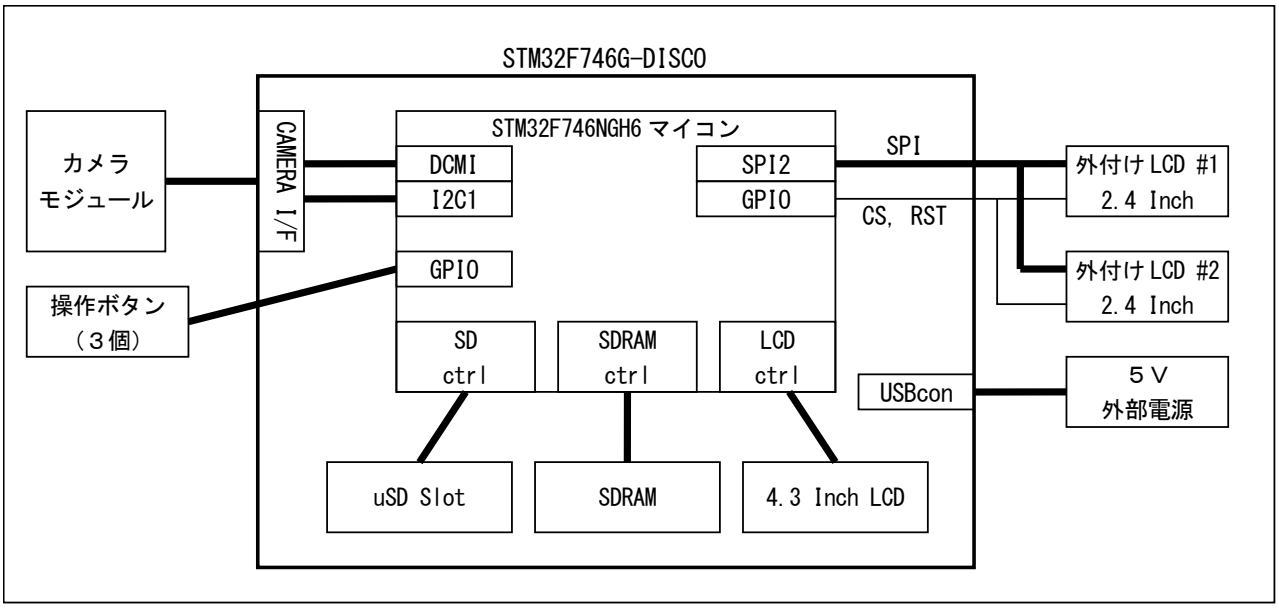
●ビルド用プラットフォーム
ubuntu 16.04-32bit 版

●クロスコンパイラ
GNU ARM Embedded Toolchain
5-2016-q3-update / Linux 32-bit
<https://developer.arm.com/-/media/Files/downloads/gnu-rm/5_4-2016q3/gcc-arm-none-eabi-5_4-2016q3-20160926-linux.tar.bz2>

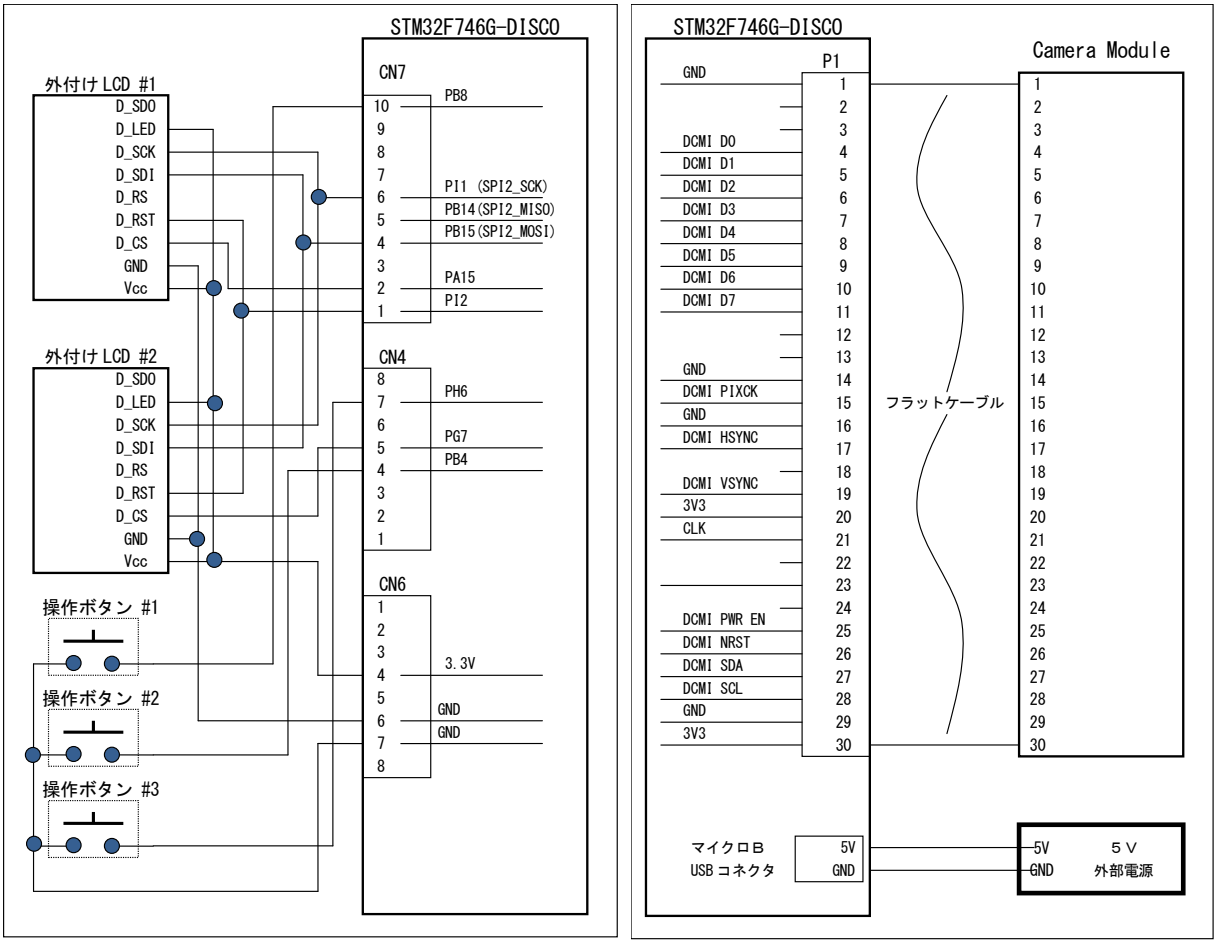
４．ハードウェアの構成・回路

ハードウェア構成と回路図を以下に記します。

●ハードウェア構成



●回路図



５．ソフトウェアの構成

●ソースツリー

```
[sample]
 ├ [appli]  ・・・・・・・・・・・・・アプリケーションディレクトリ
 │   ├ bg320x240_T1_bmp.c   ・・・・TOPPERS ロゴデータ
 │   ├ bg320x240_T2_bmp.c   ・・・・とばめロゴデータ
 │   ├ busyloop_msec.S  ・・・・・・微小時間待ちルーチン
 │   ├ imgpro_api.c   ・・・・・・・API 用コード
 │   ├ imgpro_api.h   ・・・・・・・API 用ヘッダファイル
 │   ├ imgpro_main.c  ・・・・・・・画像処理システムメインコード
 │   ├ imgpro_main.h  ・・・・・・・画像処理システムメインヘッダファイル
 │   ├ lcd_param.h  ・・・・・・・・パラメータ用ヘッダファイル
 │   ├ sample1.c  ・・・・・・・・・TOPPERS サンプルプログラム
 │   ├ sample1.cfg  ・・・・・・・・TOPPERS サンプルプログラム用定義
 │   ├ sample1.h  ・・・・・・・・・TOPPERS サンプルプログラムヘッダファイル
 │   ├ userfunc.c   ・・・・・・・・ユーザ作成関数ファイル
 │   └ userfunc_sample.c  ・・・・・ユーザ作成関数用ひな形ファイル
 ├ [CMSIS]  ・・・・・・・・・・・・ARM CIMSIS 定義ディレクトリ
 │   └ [Include]
 ├ [FatFs]  ・・・・・・・・・・・・FatFs ディレクトリ
 │   ├ 00readme.txt
 │   ├ cc932.c
 │   ├ diskio.c
 │   ├ diskio.h
 │   ├ ff.c
 │   ├ ff.h
 │   ├ ff_gen_drv.c
 │   ├ ff_gen_drv.h
 │   ├ ffconf.h
 │   ├ ffconf_template.h
 │   ├ history.txt
 │   ├ integer.h
 │   ├ main.c
 │   ├ main.h
 │   ├ sd_diskio.c
 │   ├ sd_diskio.h
 │   ├ st_readme.txt
 │   ├ stm32746g_discovery_sd.c   ・・・STMicor 提供 SD カードアクセスサンプル
 │   └ stm32746g_discovery_sd.h   ・・・STMicor 提供 SD カードアクセスサンプル
 ├ [obj]  ・・・・・・・・・・・・・・・Make 実行ディレクトリ
 │   └ Makefile
 ├ [STM32F7xx_HAL_Driver]  ・・・・・・デバイスドライバ用ディレクトリ
 │   ├ camera.h
 │   ├ device_defines.h
 │   ├ ili9328.c  ・・・・・・・・・・・外付け LCD アクセスソース
 │   ├ ov9655.c   ・・・・・・・・・・・カメラアクセスソース
 │   ├ ov9655.h   ・・・・・・・・・・・カメラアクセスソース
 │   ├ rk043fn48h.h   ・・・・・・・・・オンボード LCD アクセスソース
 │   ├ stm32746g_discovery.c  ・・・・・マイコン初期化ソース
 │   ├ stm32746g_discovery.h  ・・・・・マイコン初期化ソース
 │   ├ stm32f7xx_hal_conf.h   ・・・・・ハードアクセスコンフィグレーション
 │   ├ stm32f7xx_it.h   ・・・・・・・・内部割り込みハンドラ定義
 │   ├ [Inc]
 │   └ [Src]
 │
 └ [STM32F7xx]  ・・・・・・・・・・・・マイコン個別定義ディレクトリ
     └ [Include]
```

６．ＡＰＩ仕様

● ユーザ作成関数

画像処理システム側からユーザが作成した処理を呼び出すために利用する関数です。
userfunc_sample.c などを利用してユーザが関数とその処理を作成する必要があります。

void　userfunc_init(void)		
概要	画像処理システムの初期化時に一度だけ呼び出されるユーザが作成しなければいけない関数。 操作ボタン押下前の初期化などの処理を記述することができる。	
引数	なし	

void　userfunc_main(int btnnum)		
概要	操作ボタンが押されると呼び出されるユーザが作成しなければいけない関数。 押されたボタンに応じた画像処理や表示、保存などを行うことができる。	
引数	btnnum	押された操作ボタンの番号が渡される。

● 提供 API

ユーザ関数から画像処理システムの機能を用いるために用意した関数群です。
外付け LCD への表示や SD カードへアクセスしてファイルの読み書きを行うことができます。

void　api_copy_bmp_to_v(unsigned char *bmp, int lcdnum)		
概要	bmp で指定されたアドレスの内容を BMP データとして扱い、lcdnum で指定された外付け LCD 側のフレームバッファへ展開する。	
引数	*bmp	BMP データの先頭アドレス。
引数	lcdnum	データを展開したい側の外付け LCD の番号。

void　api_copy_fb_to_v(int lcdnum)		
概要	カメラで撮影した画像（オンボード LCD に表示されている画像）を lcdnum で指定された外付け LCD 側のフレームバッファへ展開する。	
引数	lcdnum	データを展開したい側の外付け LCD の番号。

void　api_copy_v_to_r(int lcdnum)		
概要	lcdnum で指定された外付け LCD 側のフレームバッファから出力バッファへ外付け LCD の出力形式（ＸＹ軸の変更、上位バイトと下位バイトの入替）へ変換しながら展開する。	
引数	lcdnum	データを展開したい側の外付け LCD の番号。

void　api_disp_lcd(int lcdnum)		
概要	lcdnum で指定された外付け LCD 側の出力バッファから外付け LCD へ表示を行う。	
引数	lcdnum	外付け LCD の番号。

int　api_sdcard_file_check(char *fname)		
概要	引数で指定されたファイル名のファイルが SD カード上にあるかの確認をする。	
引数	*fname	ファイル名へのポインタ。最後はヌル文字で終わること。
復帰値	int 型	0 ： 指定されたファイルがある 1 ： 指定されたファイルがない -1 ： SD カードアクセス中にエラーが発生

void api_sdcard_read(char *fname, unsigned char *buff, int size)		
概要	fname で指定されたファイル名のファイルの内容を size で指定されたバイト数分、SD カードから読み出し、buff で指定された領域へ配置する。	
引数	*fname	ファイル名へのポインタ。最後はヌル文字で終わること。
引数	*buff	読み出したファイルの内容を配置する領域の先頭アドレス。
引数	size	読み出すバイト数。

void api_sdcard_write(char *fname, unsigned char *buff, int size)		
概要	buff で指定された領域の先頭から size バイト数分を fname で指定されたファイル名で SD カードへ書き込む。	
引数	*fname	ファイル名へのポインタ。最後はヌル文字で終わること。
引数	*buff	書き込むデータが配置された領域の先頭アドレス。
引数	size	書き込むバイト数。

void api_write_bmp_file(unsigned char *photo, char *fname)		
概要	photo で指定されたアドレスを 320x240 ピクセルの RGB565 形式（フレームバッファのデータ形式）の画像データの先頭として扱い、24 ビットカラーの BMP 形式へ変換して fname で指定されたファイル名で SD カードへ保存する。	
引数	*photo	画像データの先頭アドレス。 画像データは 320x240 ピクセルの RGB565 形式とする。
引数	*fname	ファイル名へのポインタ。最後はヌル文字で終わること。

void api_camera_stop(void)	
概要	カメラの撮影を停止する。 カメラ画像利用時に停止することで新たな撮影画像で上書きされるのを抑止するのに用いる。

void api_camera_start(void)	
概要	カメラの撮影を再開する。api_camera_stop()で停止したカメラの撮影を再開するのに用いる。