Android
FlashAir
共通ライブラリ Reference

Km2Net 株式会社

目次

サンプルプログラム一覧

◎USB-IO2.0　Android サンプルプログラム一覧

プログラムベースフォルダ　　src¥AndroidStudio¥

回路図フォルダ　　src¥circuit¥UsbIoFamily¥

フォルダ	プログラム概要	回路図・設定等
UsbIoInput	USB-IO2.0 入力	
UsbIoOutput	USB-IO2.0 出力	UsbIoOutput*.png
UsbIo100V	100V 電源制御	UsbIo100V*.png

◎USB-FSIO30　Android サンプルプログラム一覧

プログラムベースフォルダ　　src¥AndroidStudio¥

回路図フォルダ　　src¥circuit¥UsbIoFamily¥

フォルダ	プログラム概要	回路図・設定等
UsbFsioAD	電圧計測	UsbFsioAD*.png
UsbFsioPwm	PWM 制御　LED サーボ	UsbFsioPwm*.png
UsbFsioSensor	センサー、液晶制御	UsbFsioSensor*.png
Pet4 ※Pet1-3 は途中説明用	ペット監視	Pet4*.png

※補足事項

AndroidStudio2.1.2 にて作成

CompileSDK　　API23

BuildToolVersion　　24.0.1

同一バージョンの AndoroidStudio が必要な場合はこちらからダウンロードしてください。

http://tools.android.com/download/studio/stable

◎FlashAir　**サンプルプログラム一覧**

プログラムベースフォルダ　　src¥ flashair¥lua¥

回路図フォルダ　　src¥circuit¥FlashAir¥

ファイル	プログラム概要	回路図
CommonDipIo.lua	DIP IO 制御ライブラリ	
CommonLcd.lua	液晶制御ライブラリ	
CommonWeb.lua	Web 利用ライブラリ	
hello.lua	入門サンプル	
led.lua	LED 表示	led*.png
temp.lua	温度監視	temp*.png
lcd.lua	液晶表示	lcd*.png
switch.lua	スイッチ監視 ※常駐アプリ	switch*.png
pet.lua	ペット監視 ※常駐アプリ	Pet4Air*.png

※補足事項

第三世代 FlashAir と DIP IO ボートを利用したサンプルです。

CONFIG は src¥flashair¥SD_WLAN 内のファイルを参考にしてください。

常駐アプリは CONFIG に自動起動設定が必要です。

USB-IO Family ライブラリリファレンス

◎クラス説明

usbiofamily-release.aar

public class UsbIoFamily extends BroadcastReceiver

USB-IOFamily を利用するためのオブジェクトを返します。

AndroidManifest.xml 追加項目

Manifest 内に追加

```
<uses-feature android:name="android.hardware.usb.host" />
```

intent-filter に下記追加

```
<action android:name=
"android.hardware.usb.action.USB_DEVICE_ATTACHED" />
```

activity に下記追加

```
<meta-data android:name=
"android.hardware.usb.action.USB_DEVICE_ATTACHED"
android:resource="@xml/device_filter" />
```

res/xml/device_filter.xml　記載内容

```
<resources>
        <usb-device vendor-id="4946" product-id="272" />
        <usb-device vendor-id="4946" product-id="273" />
        <usb-device vendor-id="4946" product-id="288" />
        <usb-device vendor-id="4946" product-id="289" />
</resources>
```

◎コールバッククラスインターフェイス説明

public interface Callbacks

コールバックメソッドをオーバーライドし、処理を記載してください。
生成したオブジェクトはUsbIoFamilyクラスのコンストラクタへ渡します。

コールバックメソッド説明

型	コールバックメソッド
void	onUsbConnect(); UsbIoFamily 接続時コールバック
void	onUsbConnectError(); UsbIoFamily 接続異常時コールバック
void	onUsbDisconnect(); UsbIoFamily 切断時コールバック

◎フィールド説明

USB 接続関連

型	フィールド
final int	MyVendorID= 0x1352; USB-IO Family のベンダーID
int	MyProductIDs[] = {0x0110,0x0111,0x0120,0x0121}; USB-IO Family のプロダクト ID
UsbManager	mUsbManager; ※USB 接続オブジェクト※
UsbDeviceConnection	mConnection; ※USB 接続オブジェクト
PendingIntent	mPermissionIntent; ※USB 接続オブジェクト※
UsbDevice	mUsbDevice; ※USB 接続オブジェクト※
UsbInterface	mIntf; Null の場合未接続 ※USB 接続オブジェクト

※Android developers の API ガイド USB Host 参照

https://developer.android.com/guide/topics/connectivity/usb/host.html

エラーコード

型	フィールド
final int	ERR_NONE = 0; エラーなし
final int	ERR_OPEN = -1; オープンエラー

final int	ERR_NOT_OPEN = -2; 未オープンエラー
final int	ERR_NOT_SUPPORT = -3; 未サポートエラー
final int	ERR_SENDRECV = -4; 送受信エラー
final int	ERR_OTHER = -9; その他エラー
final int	ERR_I2C_W_START = 0x01; I2C 書込みスタート衝突エラー
final int	ERR_I2C_W_MODE = 0x02; I2C 書込みモードエラー
final int	ERR_I2C_W_DATA = 0x03; I2C データ書込みエラー
final int	ERR_I2C_W_ACK = 0x04; I2C 書込み ACK 未確認エラー
final int	ERR_I2C_W_STOP = 0x05; I2C 書込みストップ衝突エラー
final int	ERR_I2C_R_START = 0x11; I2C 読込みスタート衝突エラー
final int	ERR_I2C_R_MODE = 0x12; I2C 読込みモードエラー
final int	ERR_I2C_R_DATA = 0x13; I2C データ読込みエラー
final int	ERR_I2C_R_ACK = 0x14; I2C 読込み ACK エラー
final int	ERR_I2C_R_STOP = 0x15; I2C 読込ストップ衝突エラー

◎コンストラクタ説明

public UsbIoFamily(Activity act, Callbacks cb) オブジェクトを生成し、USB 接続を行い、イベント発生時にコールバックメソッドを呼び出します。

◎デジタル入出力メソッド

型	メソッド
int	ctlInOut(); デジタル入出力フィールドを利用し、入出力を行い結果を返します。 戻り値　フィールド値のエラー番号

デジタル出力値クラス

public class DataOut

型	フィールド
byte	Port; ポート番号　0 の場合は未出力
void	Data; 出力値

デジタル入出力フィールド

型	フィールド
DataOut[]	dataOut = new DataOut[4]; 指定されたポートに出力
byte[]	dataIn = new byte[4]; デジタル入力時に各ポートの値を更新 添え字＋１がポート番号

◎アナログ入力メソッド

型	メソッド
int	int ctlADIn(); アナログ入力フィールドを利用し、指定されたチャネルのアナログ入力を行います。 戻り値　フィールド値のエラー番号

アナログ入力値クラス

public class DataAD

型	フィールド
byte	Chanel; アナログ入力チャネル　0 の場合は未入力
int	AD; アナログ入力値　電圧(mV)に変換するには 5000/1023 を掛ける

アナログ入力フィールド

型	フィールド
DataAD[]	dataAD = new DataAD[8]; 指定されたチャネルのアナログ値入力

◎PWM メソッド

型	メソッド
int	ctlPwmPeriod(); PWM 間隔フィールドを利用し、PWM の間隔を設定します。 戻り値　フィールド値のエラー番号
int	ctlPwmDuty(); PWM 出力フィールドを利用し、PWM の出力時間を設定します。 戻り値　フィールド値のエラー番号
int	ctlPwmOn(byte dataPwmOn); PWM の出力を ON/OFF します。 dataPwmOn　ビット ON が出力　OFF が停止　ビット位置とチャネルが対応 戻り値　フィールド値のエラー番号

PWM フィールド

型	フィールド
int[]	dataPwmPeriod = new int[5]; 66.83us 単位の PWM の間隔 添え字＋１がチャネル番号
int[]	dataPwmDuty= new int[5]; 66.83us 単位の出力時間 添え字＋１がチャネル番号

◎Ｉ２Ｃメソッド

型	メソッド
int	ctlI2cMasterOpen(byte mode); Ｉ２Ｃポートをオープンします。 mode 0:100kbps　1:400kbps 戻り値　フィールド値のエラー番号
int	ctlI2cWrite(byte addressW, byte[] data); 指定アドレスへデータを送信します。 addressW　７ビットスレーブアドレスと１ビット書込みフラグを加えた８ビット値 data　送信データ 戻り値　フィールド値のエラー番号
int	ctlI2cRead(byte addressR, byte[] data); 指定アドレスからのデータを受信します。 addressR　７ビットスレーブアドレス１ビット読込みフラグを加えた８ビット値 data　受信データ 戻り値　フィールド値のエラー番号
int	ctlI2cClose(); Ｉ２Ｃポートをクローズします。 戻り値　フィールド値のエラー番号

◎直接送受信メソッド

型	メソッド
int	ctlSendRecv(byte[] sendData, byte[] recvData) sendData を送信し、recvData に結果を受信します。送受信の内容は 64byte の通信レイアウト参照 sendData 64byte の送信データ recvData 64byte の受信データ 戻り値　フィールド値のエラー番号

◎切断メソッド

型	メソッド
int	disconnect() デバイスを切断します。 戻り値　フィールド値のエラー番号

DIP IO Lua ライブラリリファレンス

◎パッケージ　CommonDipIo.lua

DIP IO ボード制御ライブラリ

関数説明

関数	binString(num, bit)
引数	num :数値 bit:変換ビット数
戻り値	２進数文字列
説明 引数 num 指定された bit 数分下位ビットを 2 進数文字列に変換します。	

関数	boadInit(confIo)
引数	confIo: 2 ビット x 4 つの GPIO 定義 01:入力　10:出力
戻り値	なし
説明 DIP IO ボードを初期化し、GPIO0-3 の入力モードと出力モードを confIo で設定します。 例　GPIO0,1 を入力、GPIO2,3 を出力に設定する場合、2 進数で”10100101”となるため、0xA5 となる。	

関数	boadGPO(send)
引数	send:GPIO ピン出力値の送信データ
戻り値	なし
説明 DIP IO ボードの入出力を行います。 下位 4 ビットと GPIO0-3 が対応しています。	

関数	boadGPI()
引数	なし
戻り値	GPIO 入力値
説明 GPIO ピン入力値を取得し、1 バイトの数値返却します。 下位 4 ビットと GPIO0-3 が対応しています。	

関数	boadI2cW(adr, tblData)
引数	adr:I2C アドレス ＋ WriteBit (8Bit) tblData:書込みデータテーブル
戻り値	なし
説明 DIP IO ボードの I2C に 1 バイトずつデータを書込みます。 adr には WriteBit を含めた 8Bit を指定してください。	

関数	boadI2cR(adr, len)
引数	adr:I2C アドレス ＋ ReadBit (8Bit) len:読込バイト数
戻り値	読込みデータテーブル
説明 DIP IO ボードの I2C から受信したデータを読込みます。 adr には ReadBit を含めた 8Bit を指定してください。	

◎パッケージ　CommonLcd.lua

DIP IO ボード 液晶制御ライブラリ

本ライブラリ利用するには、CommonDipIo.Lua も一緒に読込んでください。

関数説明

関数	lcdInit(slaveAdr, contrast)
引数	slaveAdr:I2C アドレス ＋ W Bit (8Bit) contrast：コントラスト値(6Bit)
戻り値	なし
説明 DIP IO ボードに I2C 接続された液晶の slaveAdr へ初期化データを送信します。 slaveAdr には WriteBit を含めた 8Bit を指定してください。 contrast 参考値：電源 5V 利用時に 0x16	
関数	lcdCmd(data, afSleep)
引数	data:コマンド(1Byte) afSleep:コマンド後待ち時間(ms)
戻り値	なし
説明 指定コマンドを lcdInit で指定したアドレスに書込み、その後 afSleep(ms)スリープします。	

関数	lcdDsplay(x, y, str)
引数	x:表示桁 y:表示行 str:送信文字列
戻り値	なし
説明 液晶ディスプレイの x,y で指定された場所から文字列を表示します。 本書籍では、８×２の液晶でしたが、他のサイズの液晶も ST7032 と互換性がある制御チップ搭載なら利用可能です。	

◎パッケージ　CommonWeb.lua

FlashAir Web ライブラリ

関数説明

関数	getUrlParam(param, errVal)
引数	param:URL パラメータ errVal:取得できなかった場合の値　省略可
戻り値	パラメータ値
説明 param に指定された URL パラメータの値を返します。 パラメータが取得できなかった場合は、errVal 値を返します。	

関数	printBR(str)
引数	Str:文字列
戻り値	なし
説明 指定された文字列に、” ”を付加し print します。	