ADK互換モジュールで遊ぶAndroid™

第4回名古屋Android勉強会
2012/2/18
愛知工業大学　本山キャンパス
@magoroku15　最底辺活動家

このコースの目的と内容

目的

AndroidアプリケーションデベロッパがAndroid Open Accessory を理解する。同時にHWを理解する事が広げる世界を体感する。

内容

1. 互換モジュールの組み立て
2. Androidとの接続と動作確認
3. 互換モジュールの仕組み
4. Android側の仕組み

PART0
事前準備

最新リソース

- このドキュメントを含む、最新のリソースは以下で管理しています
  - https://github.com/magoroku15/OpenAccessoryDemo
    - App Source CodeAndroid　アプリのソースコード
    - Doc　ドキュメント類
    - Firmware　マイコンのソース・バイナリ
- 紹介するMicrochip Technology Inc作成のアンドロイドアプリをAndroid Marketからインストールしておいてください
  - Microchipで検索
    - 基本的なアクセサリデモ3.x
    - 基本的なアクセサリデモ2.3.x以降

レベルごとの準備

- レベル１　手ぶらコース
  - 特に準備は不要です
- レベル２　実機に接続して動かす
  - ADKに対応したAndroid実機とACアダプタを用意
- レベル３　マイコン実行イメージのビルドまで
  - MPLABXをインストールしたPCを用意
- レベル４　Androidアプリのビルドまで
  - Android SDKをインストールしたPCを用意

ソースコード・ドキュメント

- Githubで管理

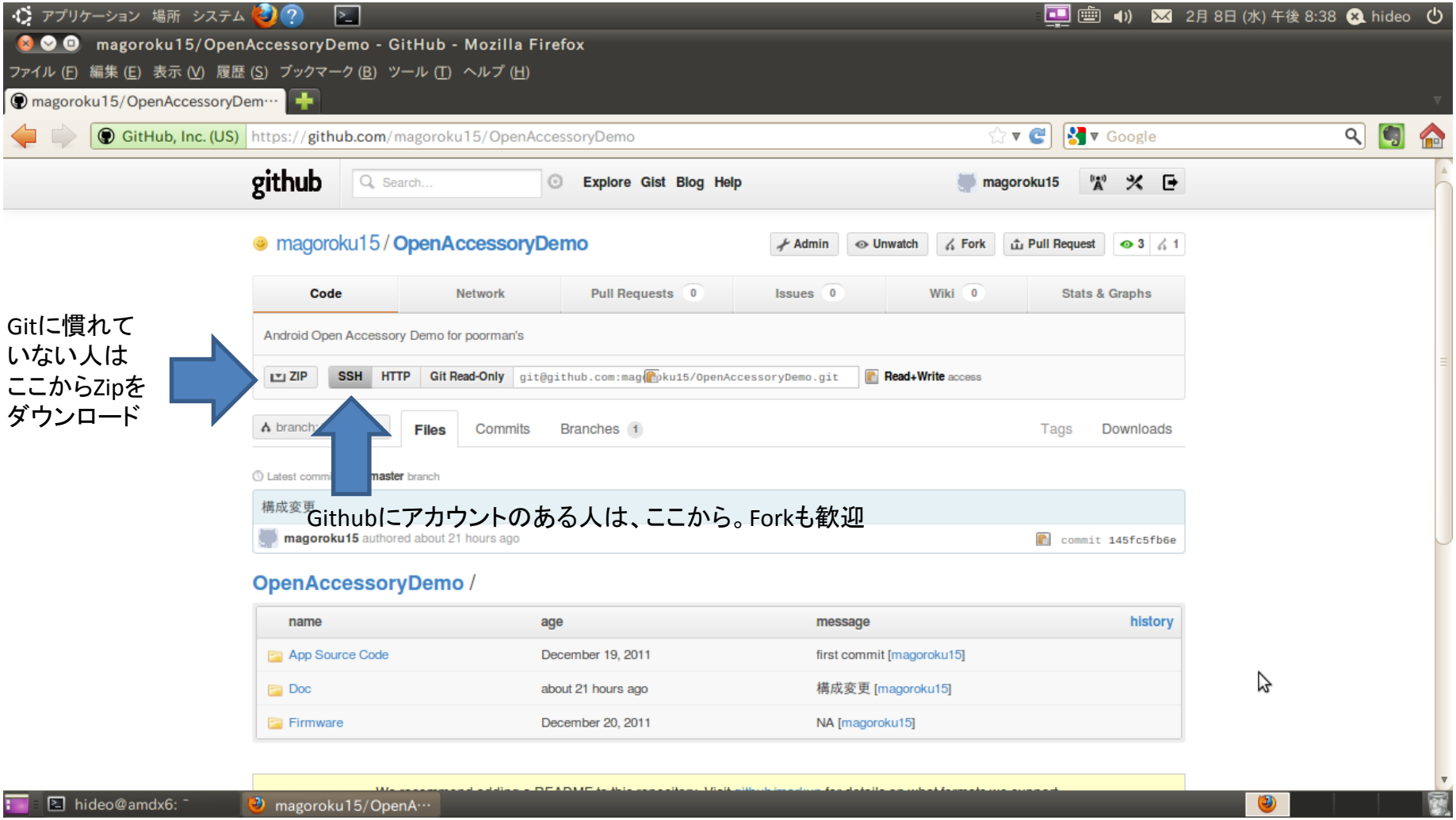
  https://github.com/magoroku15/OpenAccessoryDemo

- ２つの方法

  A) Githubにアカウントを作ってfork
    - Linux/Macで開発する人にお勧め
    - Zipでダウンロード改変してcommit & push

  B) ZIPアーカイブをダウンロード
    - Windowsで開発する人向け

https://github.com/magoroku15/OpenAccessoryDemo



Gitに慣れていない人はここからZipをダウンロード

Githubにアカウントのある人は、ここから。Forkも歓迎

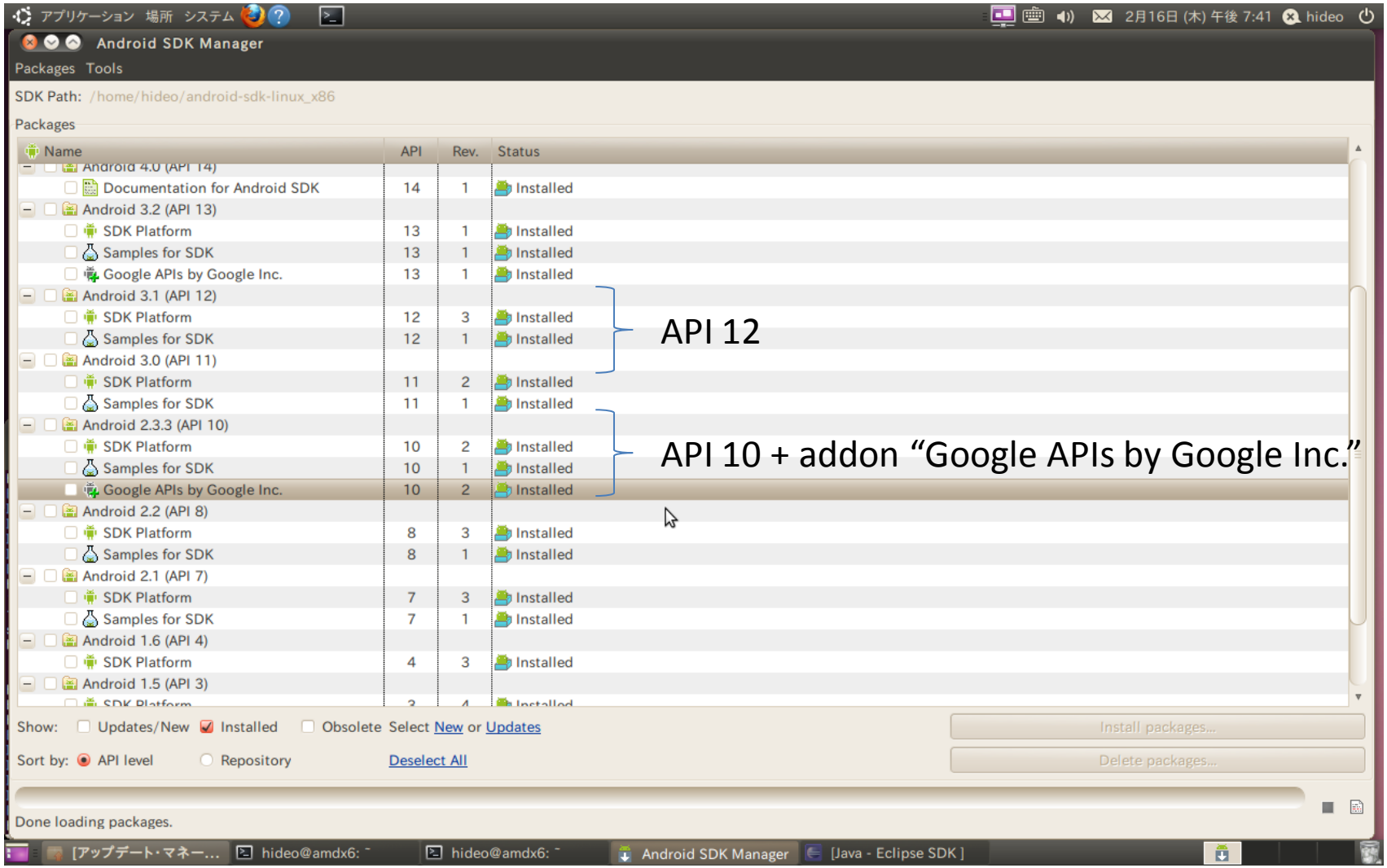
OpenAccessoryDemoの内容

- App Source Code　　Androidアプリのソース
  - v2.3.x　　2.3.4以降のADKアプリのソース
  - V3.x　　3.x以降のADKアプリのs-す
- Doc
  - Poorman’s adk.pptx　　この資料
  - OpenAccDemo.pdf　　回路図　PDF形式
  - OpenAccDemo.sch　　回路図　EagleCAD形式
- Firmware　　Firmwareのソース

Android SDKのインストール

- ハンズオンでAndroidのサンプルプログラムの解説を行い、その中でAppの作成に軽く触れます
- 動作確認はマーケットからダウンロードしたアプリで行います
- サンプルソースのバージョンに合わせて各自で事前にインストールを済ませておいてください
  - Androidのバージョンが2.3.xの場合
    - API 10に加えてGoogle APIアドオンが必要
  - Androidのバージョンが3.1以降の場合
    - API12が必要

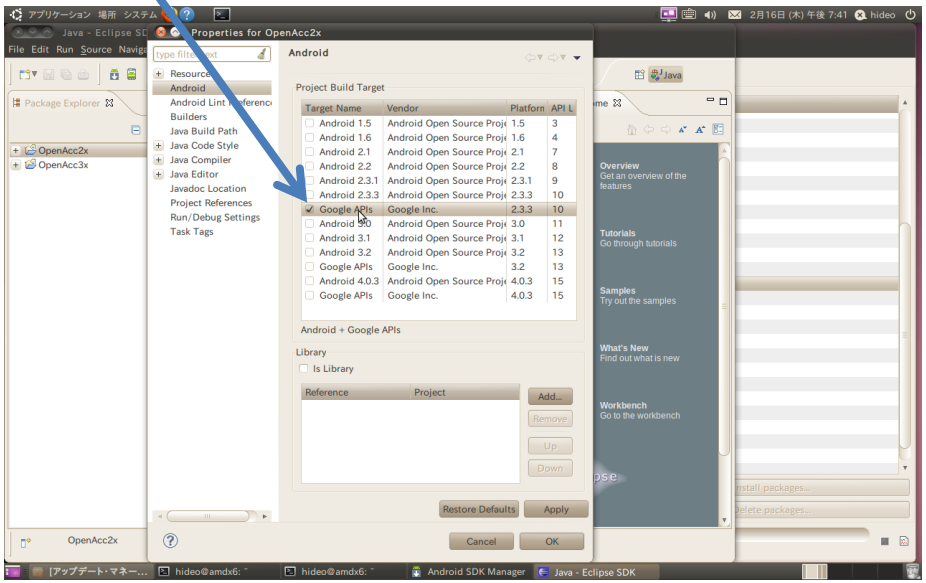
SDKのバージョン・アドオン



Android SDKの確認　v2.x

- v2.3.xの場合

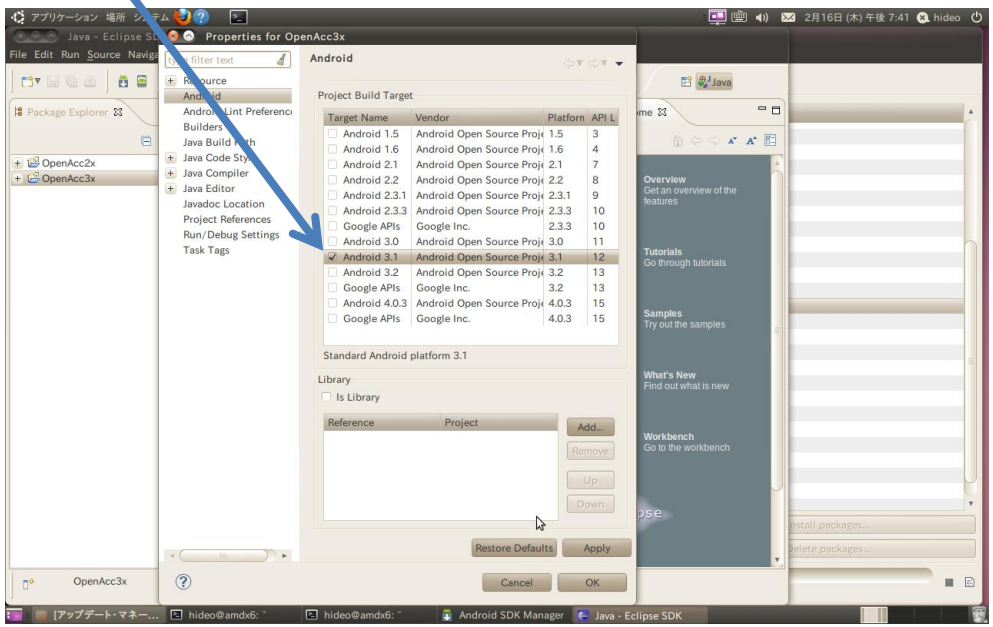
0) SDK Managerで「Android 2.3.3 (API10)のGoogle APIsアドオン」をインストール
1) 新規のAndroidプロジェクトを作成
2) 「Create project from existing source」を選択
3) Locationをgithubからダウンロードしたzipを展開したフォルダの中の「OpenAccessoryDemo/App Source Code/v2.3.x」に設定
4) Nextをクリック
5) Build Targetを「Google APIs-2.3.3-10」
6) Finishをクリック



Android SDKの確認 v3.x,v4.x

- v3.x,v4.xの場合

0) SDK Managerで「Android 3.1 (API12) SDK Platform」をインストール
1) 新規のAndroidプロジェクトを作成
2) 「Create project from existing source」を選択
3) Locationをgithubからダウンロードしたzipを展開したフォルダの中の「OpenAccessoryDemo/App Source Code/v3.x」に設定
4) Nextをクリック
5) Build Targetを「Android 3.1 (API12)」
6) Finishをクリック

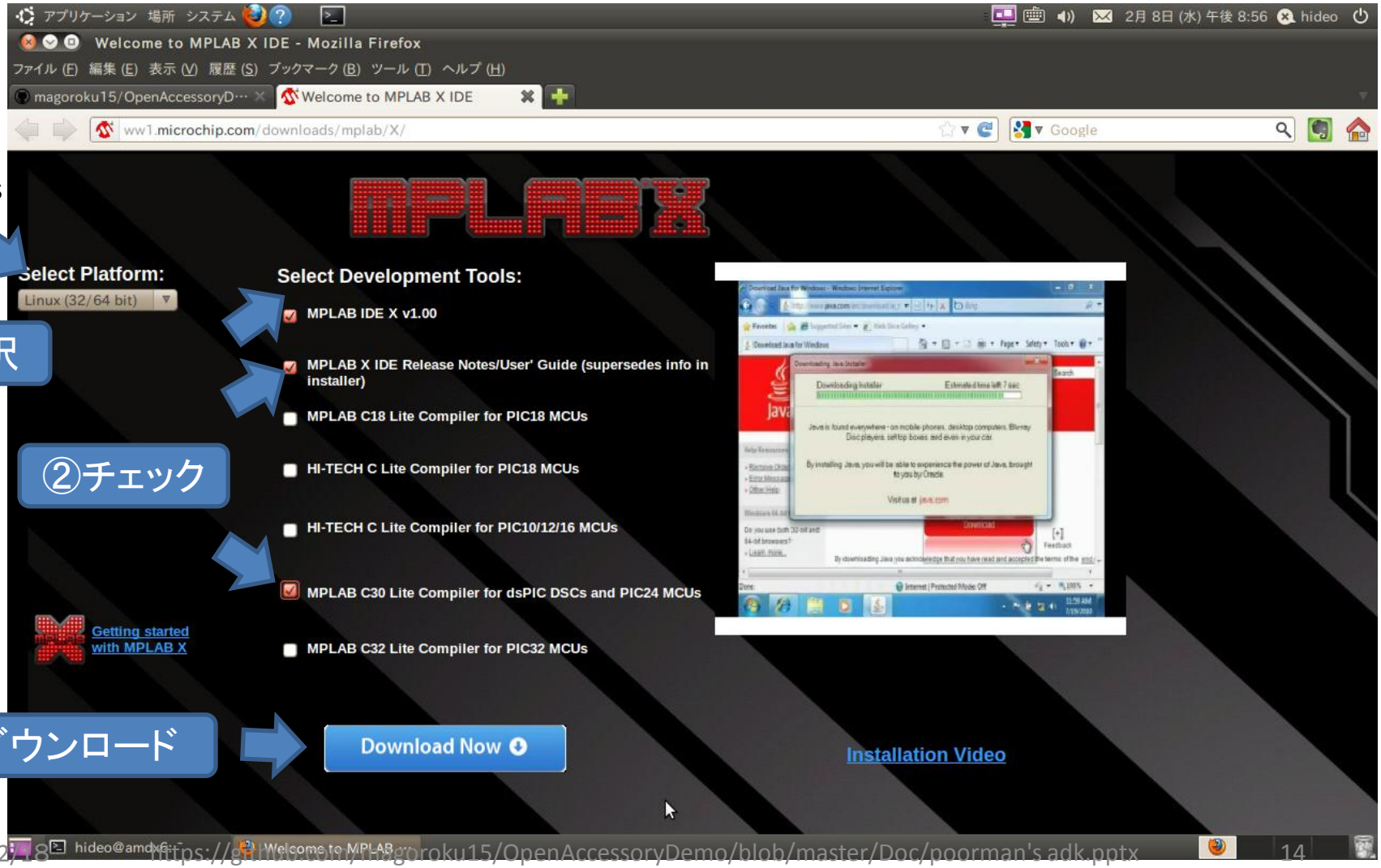


MPLABXのインストール

- PIC用のIDE
  - 従来のMPLABはWindows専用だった
  - MPLABXはNetBeansベースでマルチプラットフォーム
  - PIC24シリーズ向けはコンパイラも無償/最適化に制限
- http://ww1.microchip.com/downloads/mplab/X/から
  1. Platformを選択
  2. 以下をチェックして[Download Now]
     - MPLAB IDE X v1.00
     - **MPLAB X IDE Release Notes/User' Guide (supersedes info in installer)**
     - **MPLAB C30 Lite Compiler for dsPIC DSCs and PIC24 MCUs**
  3. [Download Now]

MPLABXのインストール

Mac
Linux
Windows



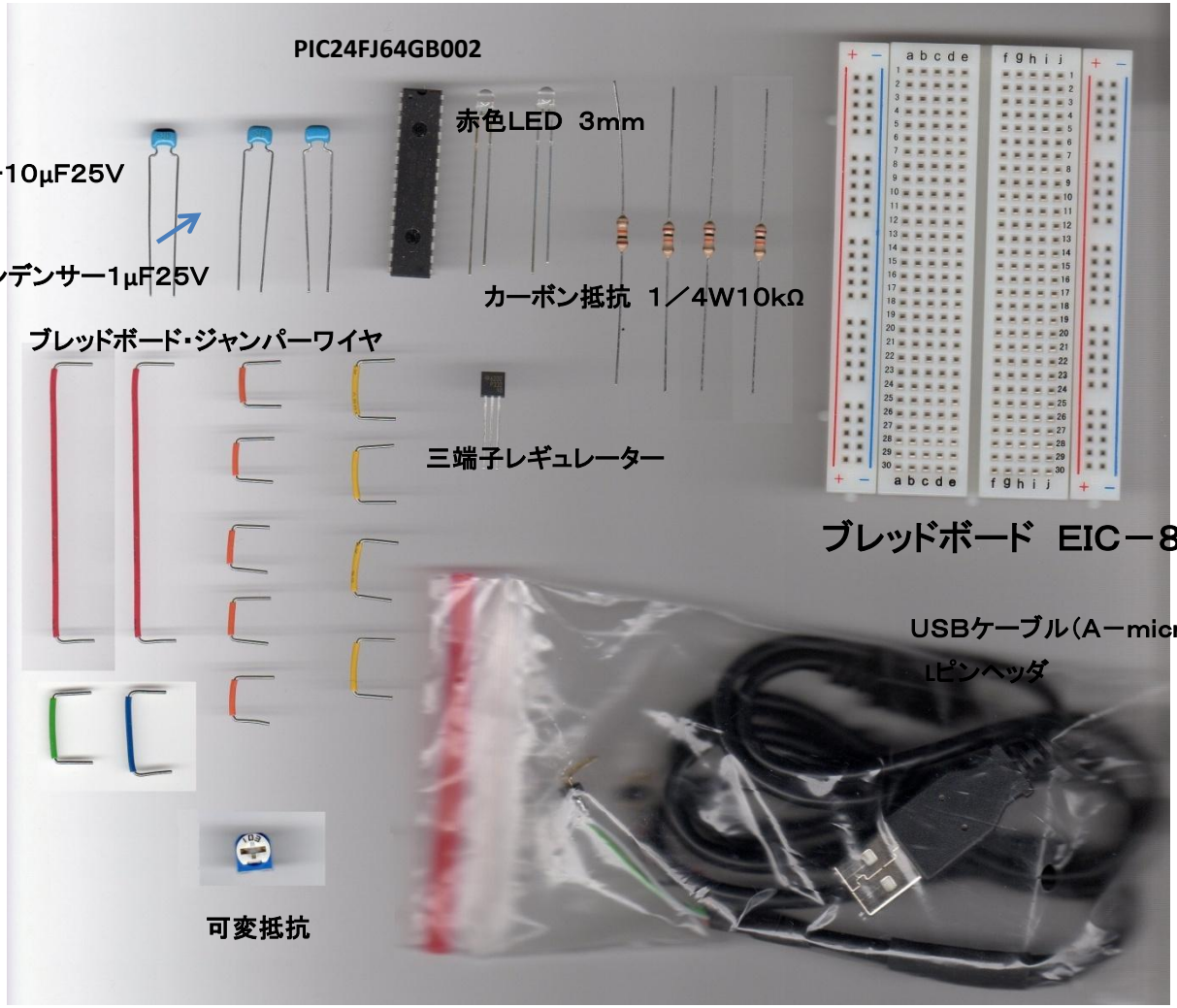
①選択

②チェック

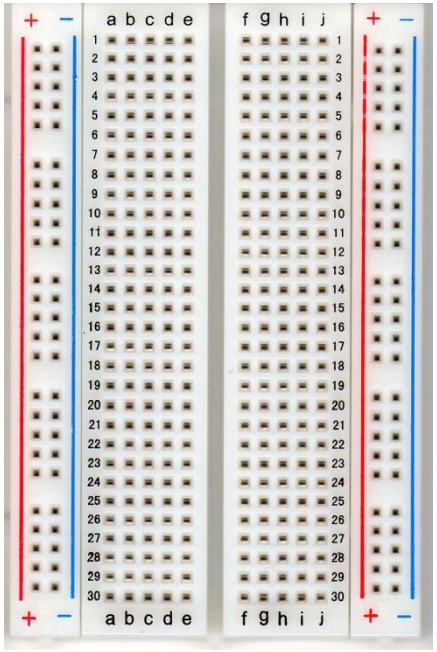
③ダウンロード

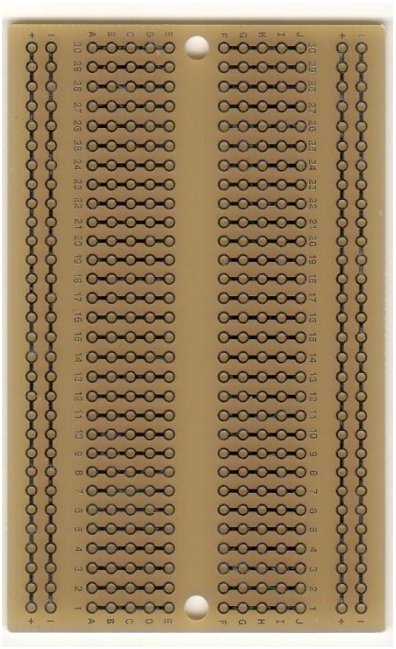
PART1
互換モジュールの組み立て

配布部品


PIC24FJ64GB002
赤色LED 3mm
積層セラミックコンデンサー10μF25V
積層セラミックコンデンサー1μF25V
カーボン抵抗 1／4W10kΩ
ブレッドボード・ジャンパーワイヤ
三端子レギュレーター
ブレッドボード EIC－801
USBケーブル（A－microB）
Lピンヘッダ
可変抵抗

ブレッドボード

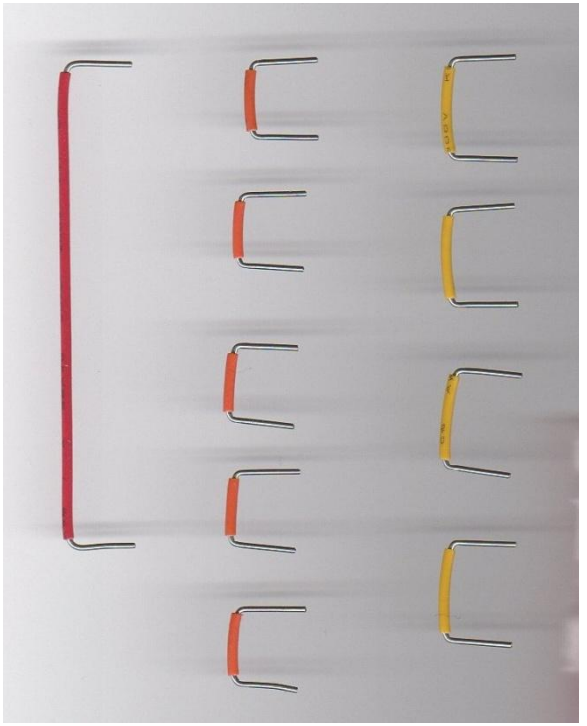




内部の配線

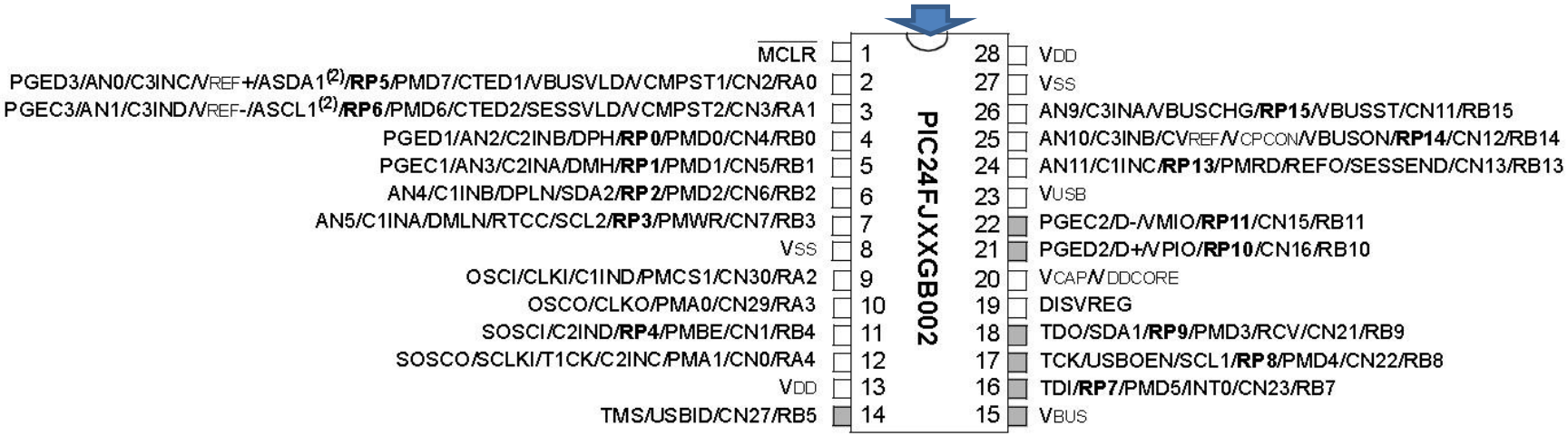
ジャンパ

- ブレッドボードに刺して回路を構成する線



マイコン PIC24FJ64GB002

- Microchip社の16bitマイコン max 32MHz
  - 64Kbyte Program Memory (Flash)、64Kbyte RAM
  - I²C, IrDA, SPI, UART/USART, USB OTG
- ハンズオン用プログラムは書き込み済で配布



ピン配置

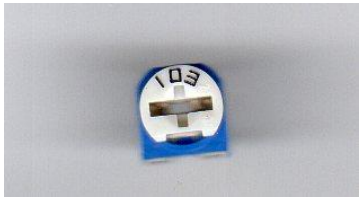
抵抗

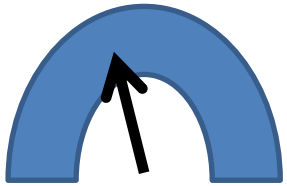
- 電流の流れを抑止する
- 単位はΩ(オーム)
  - 抵抗の値が小さいと導体
  - 抵抗の値が大きいと絶縁体に近づく
- 極性なし

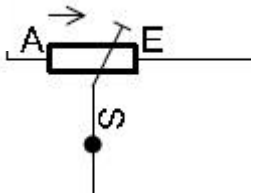
回路記号

ポテンションメータ/可変抵抗

- 抵抗値を変化させる
- 単位はΩ(オーム)
  - 音響装置の「ボリューム」がこれ
  - 回転角・位地の検出にも使う



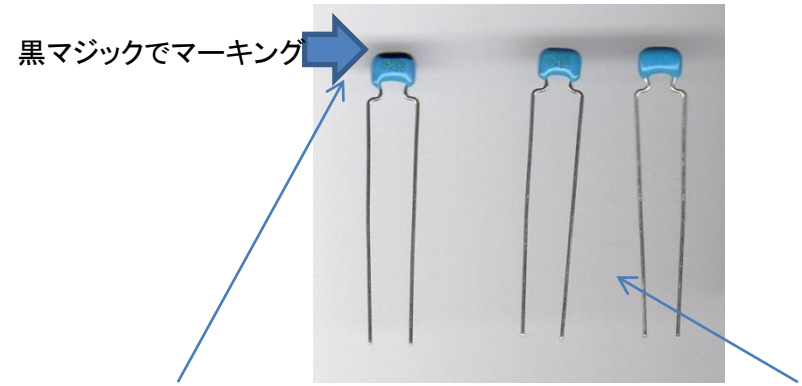




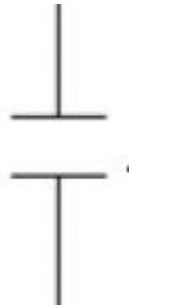
回路記号

コンデンサ

- Capacitor(キャパシタ)とも言う
- 単位はF(ファラッド)
- 微量の電気を蓄える
  - 直流は流れない
  - 交流は流れやすい
- 極性の有無に注意
  - 今回の積層セラミックコンデンサは極性なし



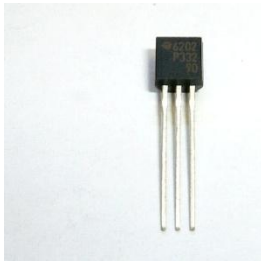
積層セラミックコンデンサー10μF25V　　積層セラミックコンデンサー1μF25V

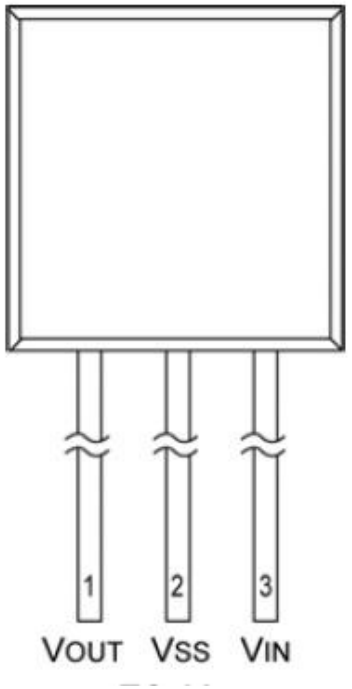


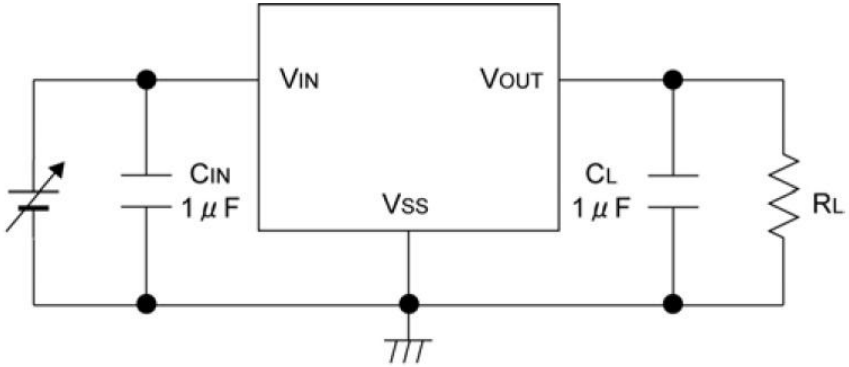
回路記号

3端子レギュレータ

- 電圧を一定に保つ機能を備えたIC
- 5V(充電用ACアダプタ)から3.3Vを生成



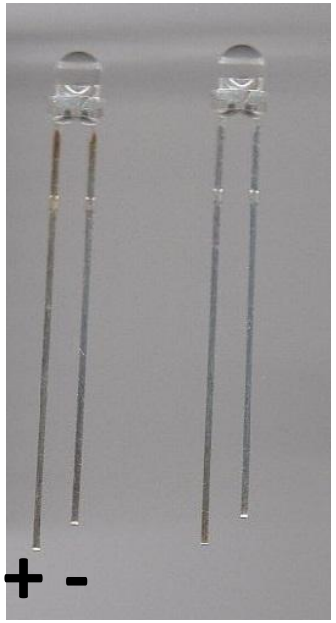


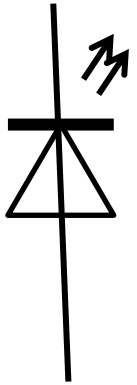


回路図(周辺込み)

LED

- 発光ダイオード
  - **LED**（エルイーディー: **Light Emitting Diode**）
  - 極性あり、一方向にしか電流を流さない



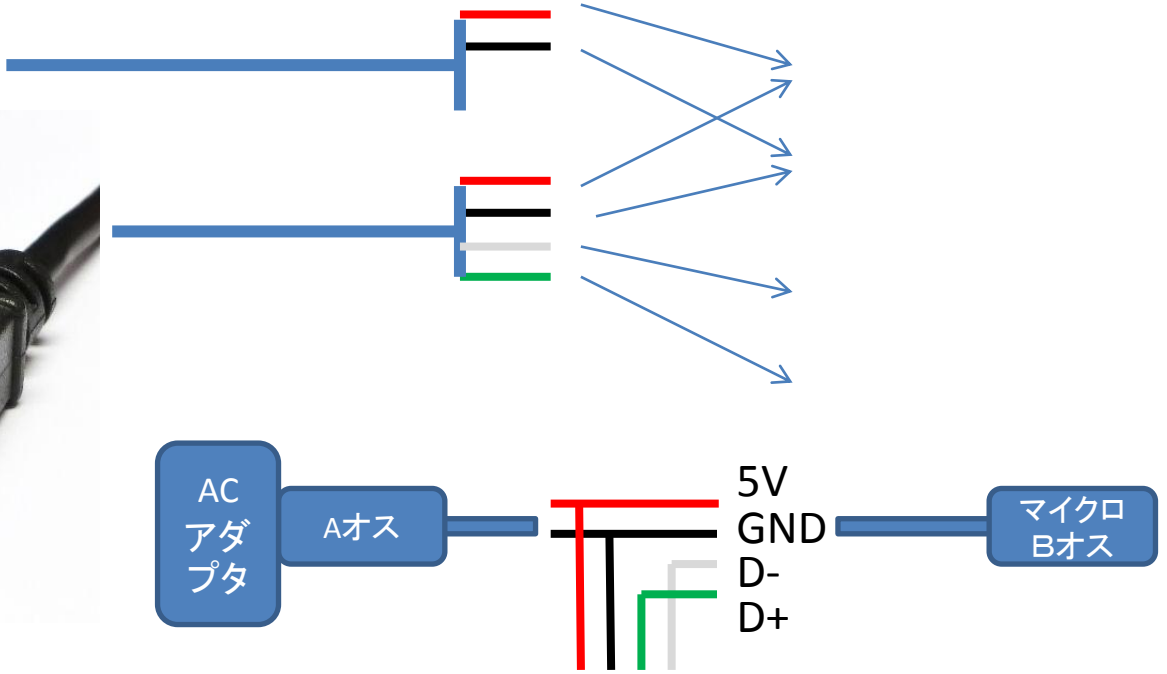


回路記号

USBケーブル

- Aオスマイクロ Bオスを改造（改造済で配布）
  - ACアダプタから5Vを取り、マイクロBオスに回す
  - マイクロBのデータ線を取りだす

Aオス－マイクロBオス



互換モジュールの組み立て

- 利用する資料
  - 回路図
  - 実体配線図
  - 組み立て手順

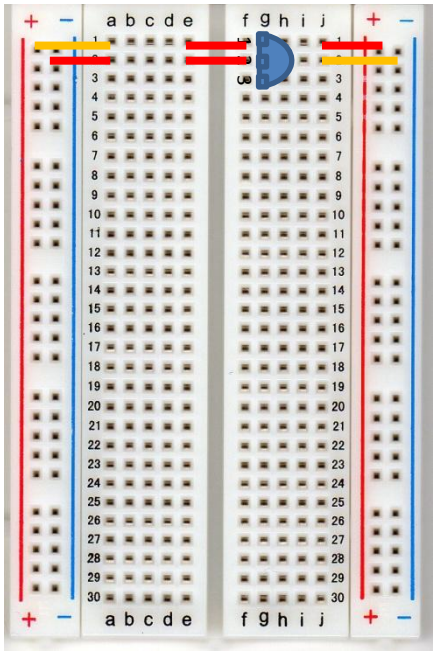
配布部品と入手価格

poorman's ADK 30台分の発注部品

			購入単位			1台単位		
購入先	通販型番	発注量	個数	価格	単価	個数	小計	
秋月	C-05223	30	1	130	130	1	130	USBケーブル（A－microB）
秋月	P-02315	30	1	250	250	1	250	ブレッドボード・ジャンパーワイヤ
秋月	P-00315	30	1	250	250	1	250	ブレッドボード　EIC－801
秋月	I-03421	15	2	100	50	1	50	三端子レギュレーター[3．3V] XC6202P332TB
秋月	P-05105	60	1	15	15	2	30	積層セラミックコンデンサー1μ F50V
秋月	P-05103	30	1	30	30	1	30	積層セラミックコンデンサー10μ F25V
秋月	I-00562	1	100	350	3.5	2	7	赤色LED　3mm
秋月	R-25103	2	100	100	1	4	4	カーボン抵抗　1／4W10kΩ
秋月	C-01627	10	5	20	4	1	4	Lピンヘッダ1x20
秋月	P-02470	4	10	200	20	1	20	半固定抵抗10kΩ
秋月	P-04089	1	100	500	5	1	5	チャック袋
秋月	送料2回分	1	30	1000	33	2	67	送料500円x2回
digikey	703-7602	1	30	9821	327	1	327	PIC24FJ64GB002

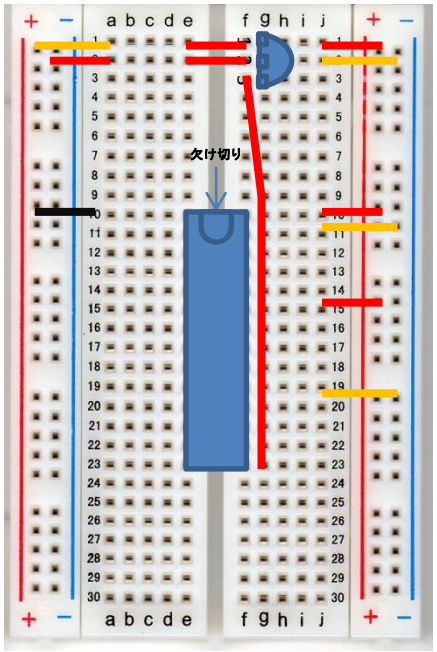
合計　1174

Step1　電源



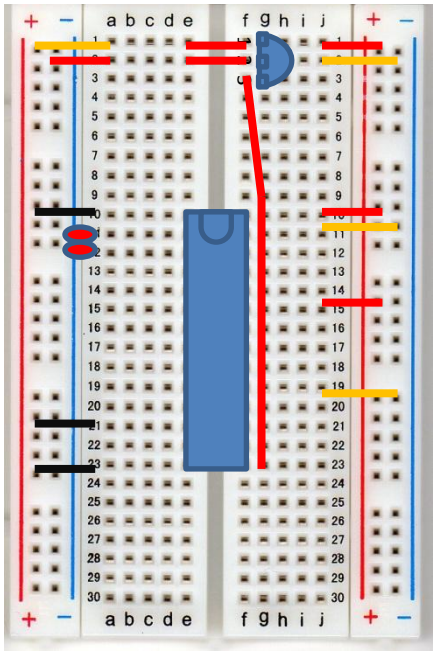
1. g1にレギュレータの1
2. g2にレギュレータの2
3. g3にレギュレータの3
4. ―を4か所
5. ―を2か所
6. 積層セラミックコンデンサー1μF25V2個
   - h1-h2
   - h2-h3

Step2　マイコン



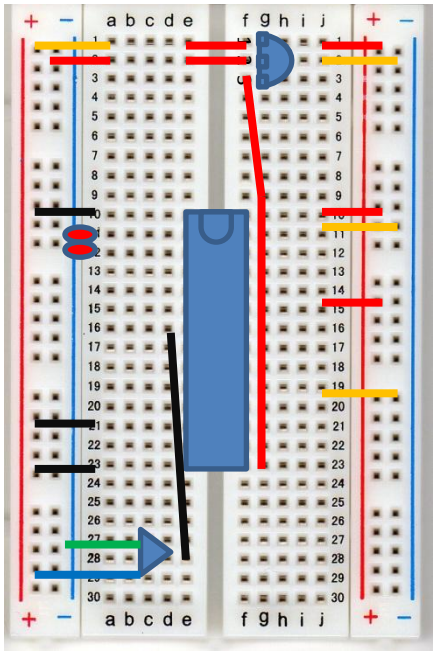
1. e10-e23-f10-f23’にマイコン
2. ー をf3-g23へ
3. ー をj10-ー へ
4. ー をj11-＋へ
5. ー をj15-ー へ
6. ー をj19-＋へ
7. 積層セラミックコンデンサー10μF25をj18-ー へ
8. 抵抗10kΩをa10- ＋へ

Step3　LED/SW



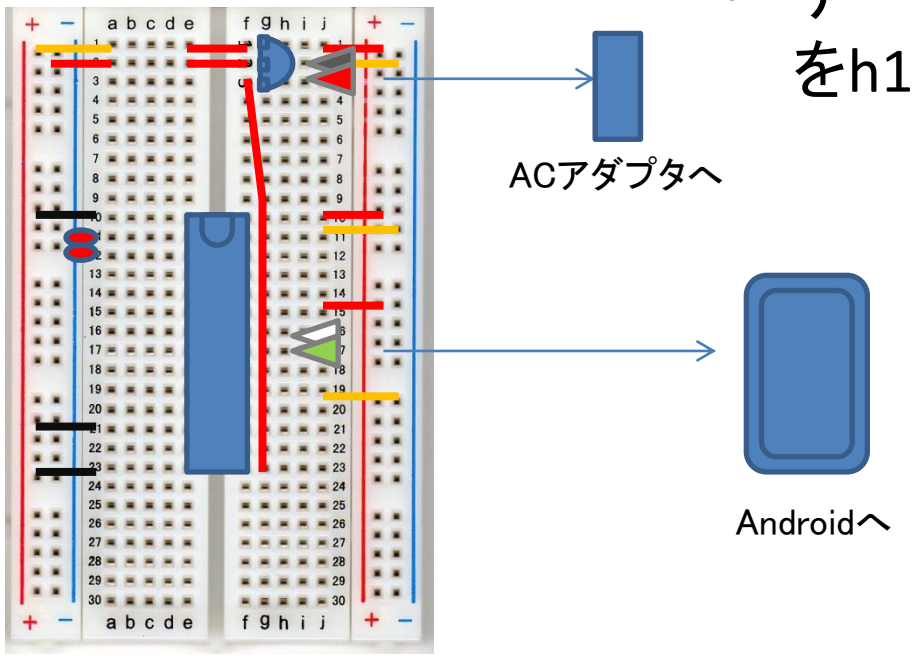
- LED、足の長い方をa11,短い方を－へ
- LED、足の長い方をa12,短い方を－へ
- 抵抗10kΩ、a21-＋へ
- 抵抗10kΩ、a23-＋へ

Step4　可変抵抗



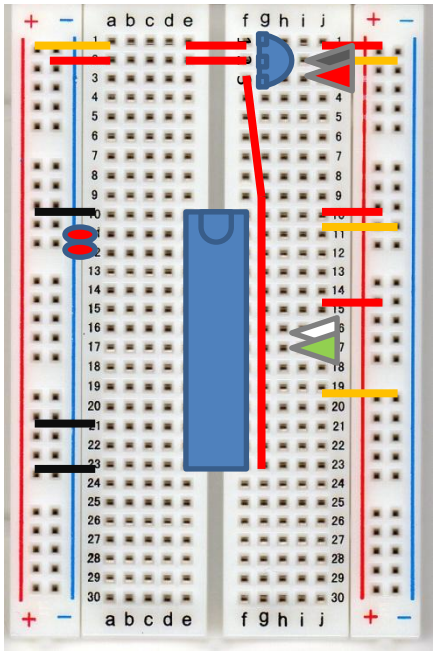
- 組み付け順注意
- d16-e28に抵抗を
- ー(緑)をc27-ーに
- ー(青)をc29-＋に
- 可変抵抗をc27-c29-e28へ

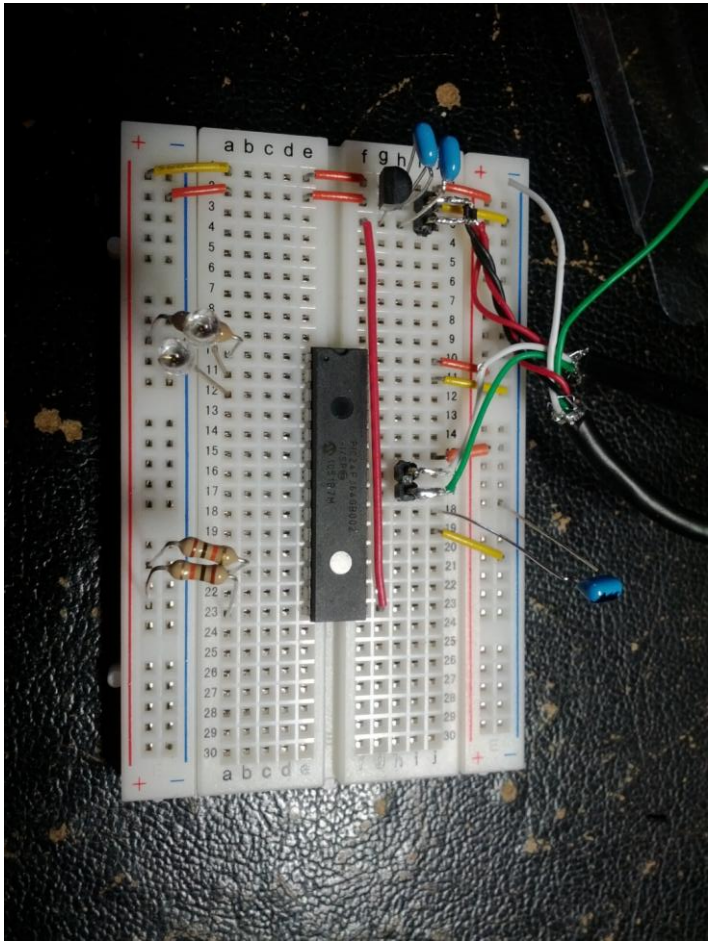
Step5　USBケーブル



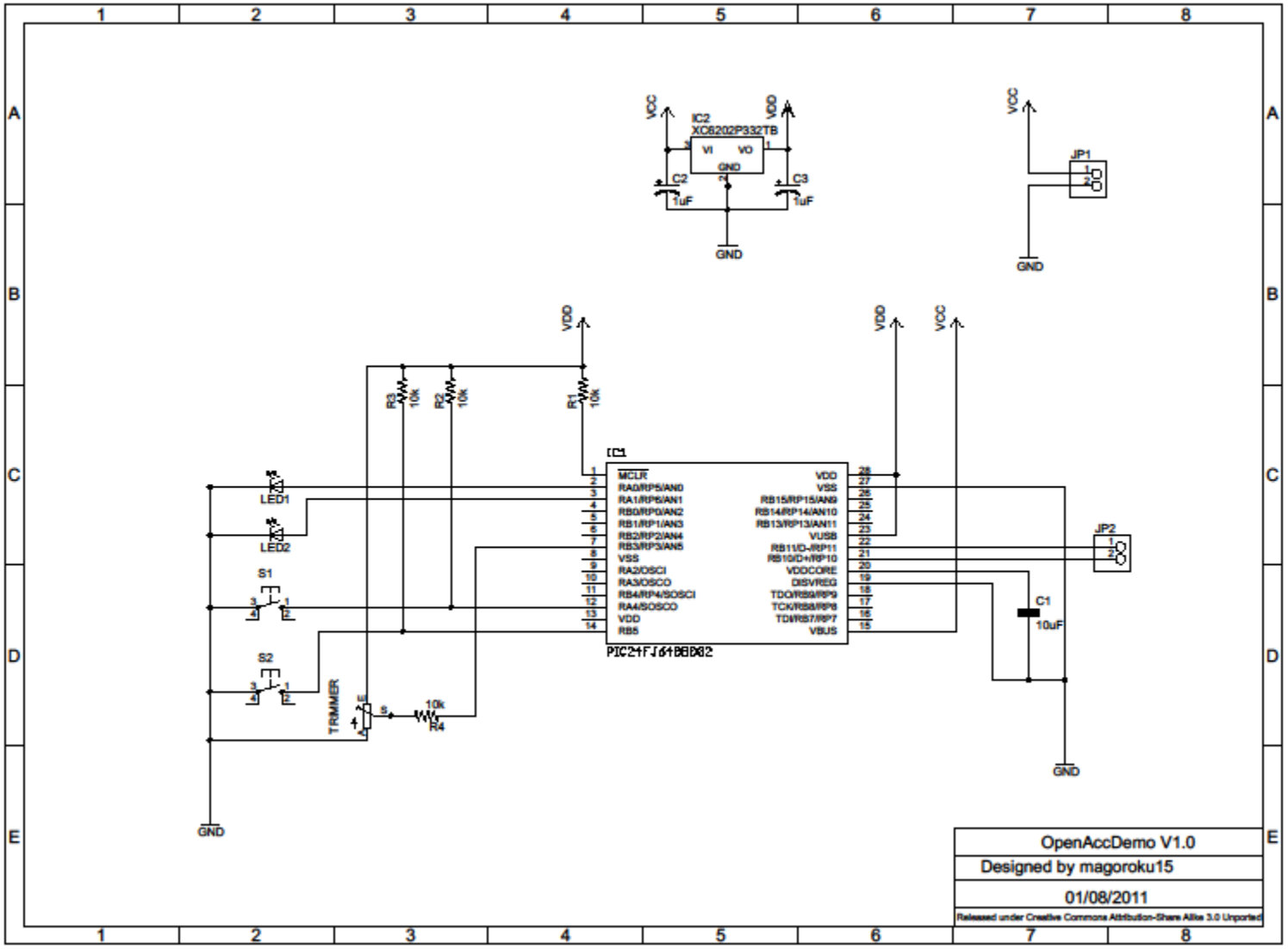
- 電源、黒をj2,赤をj3
- データ 白(D-)をh16, 緑(D+)をh17

完成図と写真





回路図


IC2
XC6202P332TB
VI
VO
GND
C2
1uF
C3
1uF
VCC
VDD
JP1
JP2
R1
10k
R2
10k
R3
10k
R4
10k
LED1
LED2
S1
S2
TRIMMER
IC1
PIC24FJ64GB002
MCLR
RA0/RP5/AN0
RA1/RP6/AN1
RB0/RP0/AN2
RB1/RP1/AN3
RB2/RP2/AN4
RB3/RP3/AN5
VSS
RA2/OSCI
RA3/OSCO
RB4/RP4/SOSCI
RA4/SOSCO
VDD
RB5
VDD
VSS
RB15/RP15/AN9
RB14/RP14/AN10
RB13/RP13/AN11
VUSB
RB11/D-/RP11
RB10/D+/RP10
VDDCORE
DISVREG
TDO/RB9/RP9
TCK/RB8/RP8
TDI/RB7/RP7
VBUS
C1
10uF
OpenAccDemo V1.0
Designed by magoroku15
01/08/2011
Released under Creative Commons Attribution-Share Alike 3.0 Unported

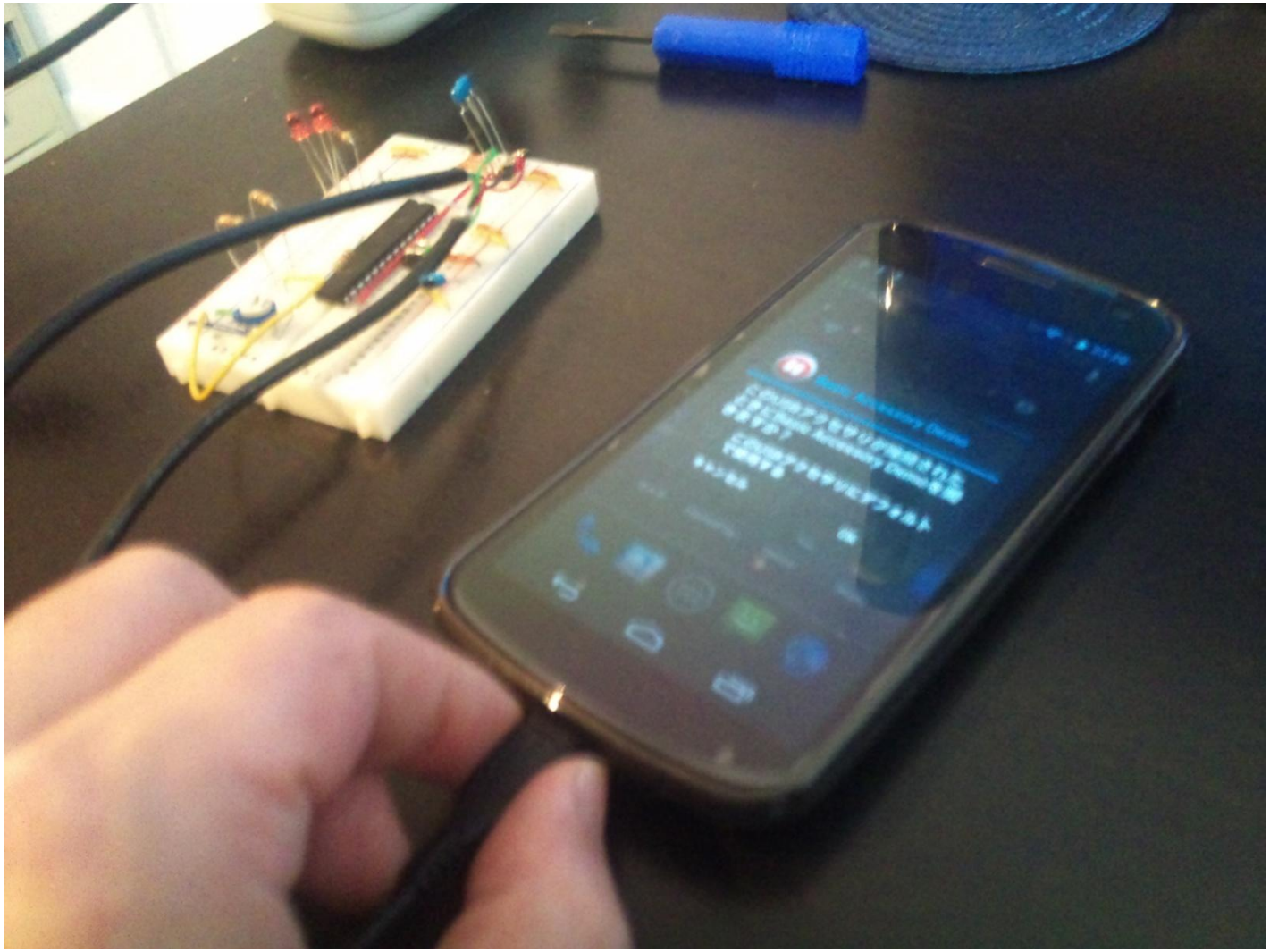
PART2
ANDROIDとの接続と動作確認

Step1 アプリのインストール

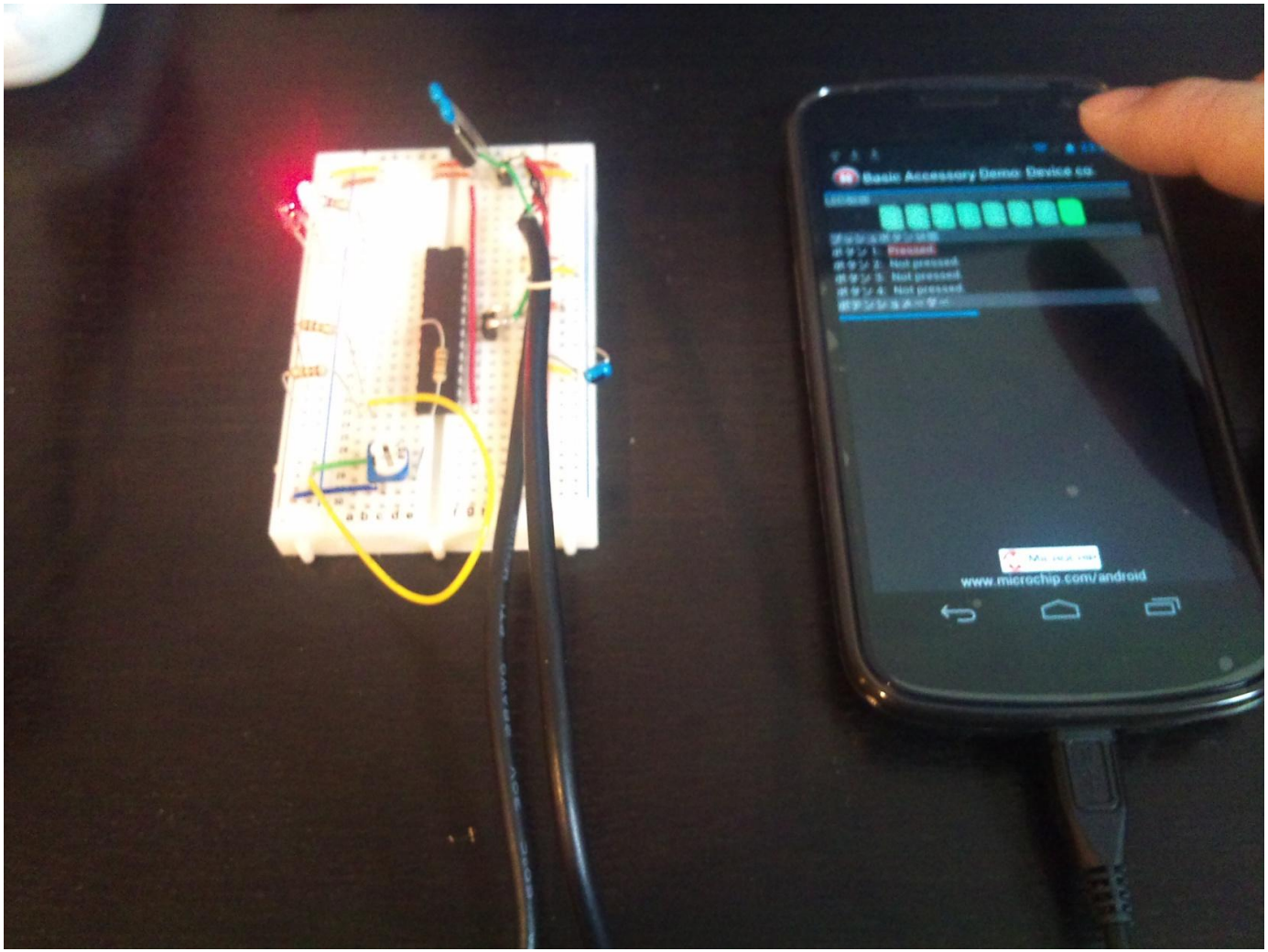
- Android Marketからインストール
  - Microchipで検索
    - 基本的なアクセサリデモ3.x
    - 基本的なアクセサリデモ2.3.x以降

Step2 ACアダプタに接続

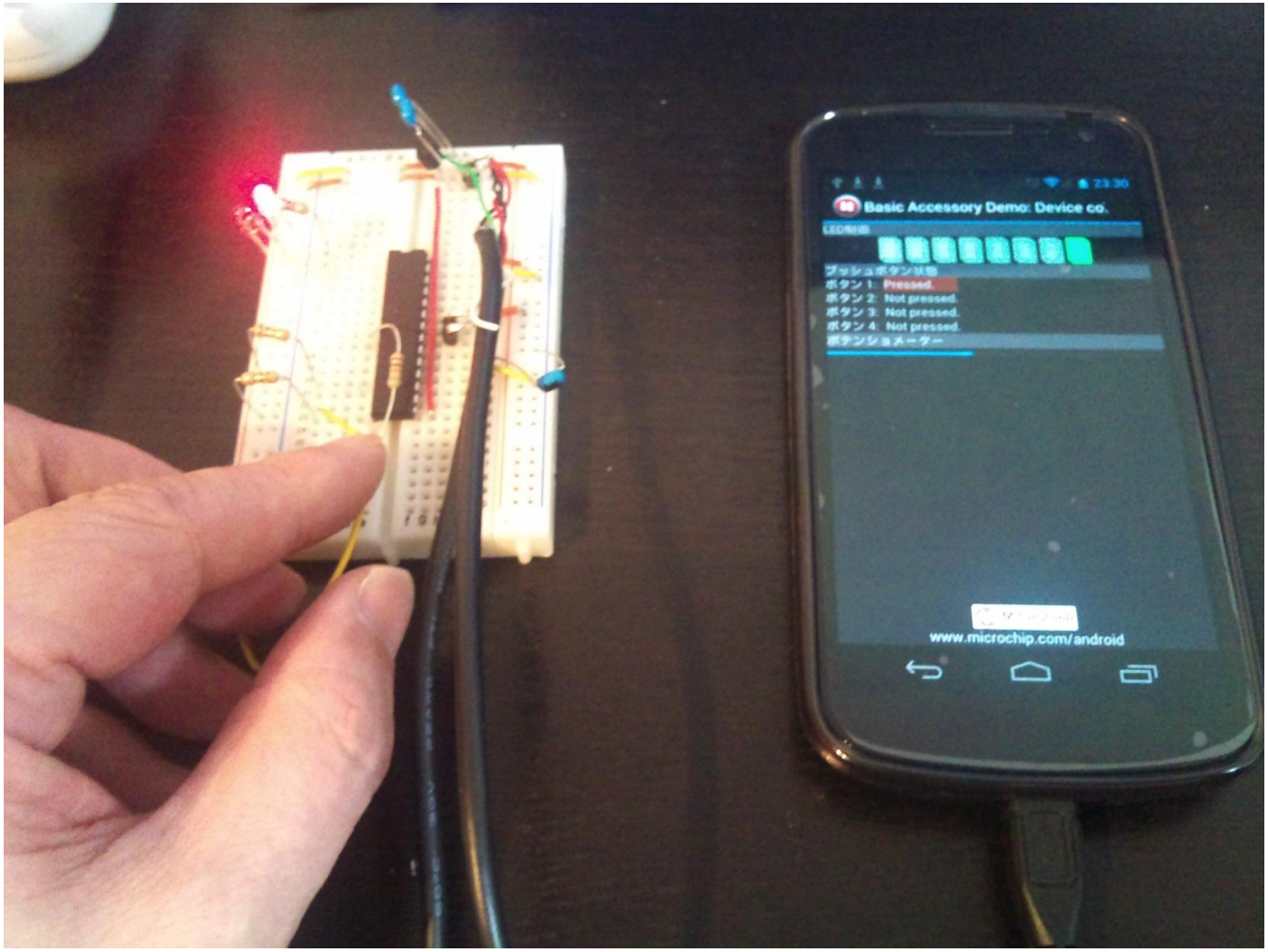
Step3 Androidに接続



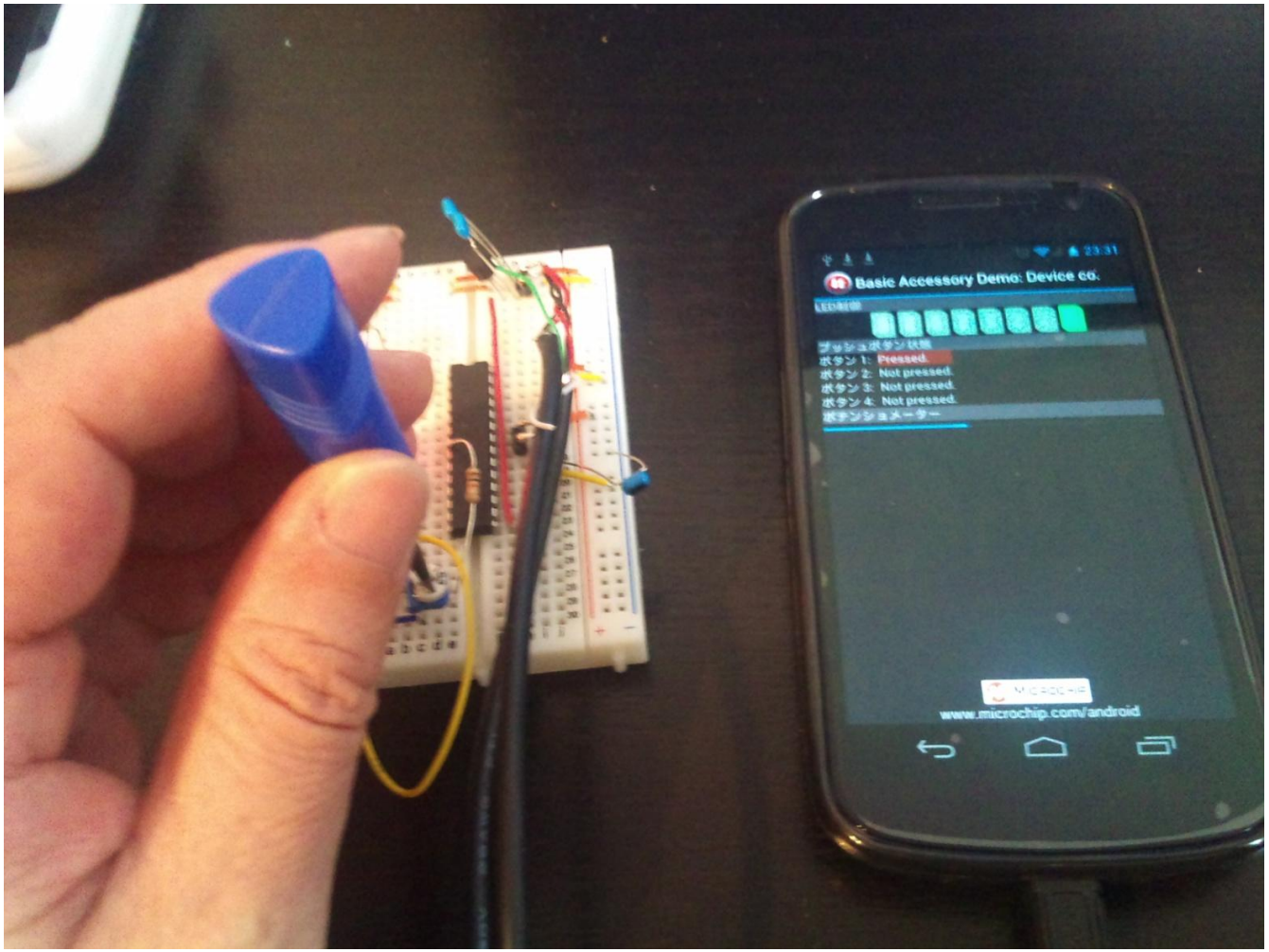
LEDを点灯



スイッチを操作


Basic Accessory Demo: Device co.
ボタン 1: Pressed.
ボタン 2: Not pressed.
ボタン 3: Not pressed.
ボタン 4: Not pressed.
www.microchip.com/android

ポテンションメータを操作



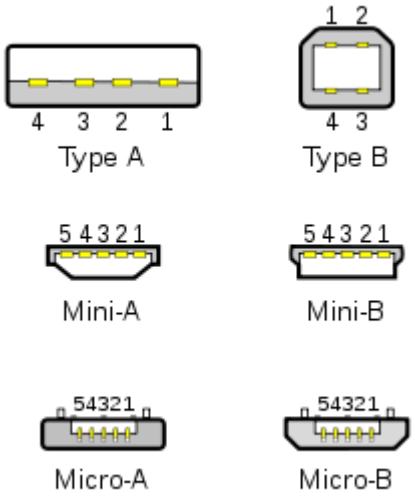
PART3
USBとOPEN ACCESSORY

USB　ホスト・ディバイス

- 基本は高速のシリアル通信
- ４線式
  - 通信はディファレンシャル D+,D-
  - 給電機能を持つ　5V, GND
- ホスト
  - 一般的にPC側
  - マスタとして動作
  - 複数のディバイスを収容・管理
  - 複雑なソフトウエアスタック
- ディバイス
  - 一般的に装置側、マウス、KBD、プリンタ、USBメモリ
  - スレーブとして動作
  - 単純な機能

USB　コネクタ

- 4線式　　1: +　2: D-　3: D+　4: -
  - ホスト側　　Type A
  - ディバイス側　　Type B
- 5線式　　1: +　2: D-　3: D+　4: ID　5:　-
  - IDでホスト、ディバイスを判別



Wikipediaより

Androidでの利用例

- ホスト
  - システム側での対応は限定的、マウス対応(ICS)など
  - アプリ側でUSB managerと連携
  - ドライバ相当の動作をアプリとして記述・配布
- ディバイス
  - PCがホスト、Androidがディバイス
  - ADB, Mass Storage
- Open Accessory
  - PCがホスト、Androidがディバイス
  - ADBに類似
  - システムが利用していたエンドポイントをアプリに開放

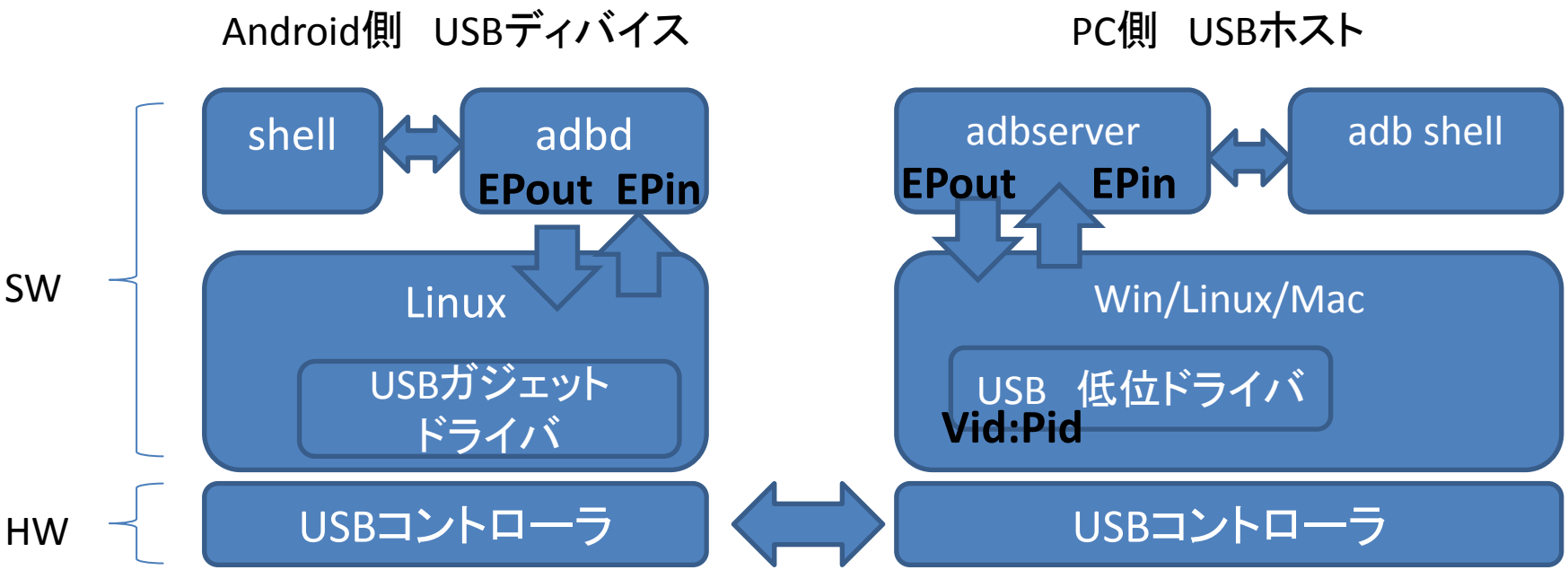
ADBのしくみ

- Android:ドライバレイヤ
  - ADB用のVid, Pidを登録してエンドポイントを作成
- Android:アプリレイヤ
  - システム起動時にadbdコマンドを起動
- PCに接続すると
  - 特定のVid, Pidを示し、ADBの接続をPC側が認識
  - 専用Endpointがみえるのでこれをadbコマンドが握る

PC adbコマンド⇔PC バルク転送⇔Android バルク転送 ⇔ /dev/adb ⇔ adbd

ADBの構成

- $adb shellを実行した場合のデータパス



ADBのID定義

- C:¥Program Files¥Android¥android-sdk-windows¥extras¥google¥usb_driver¥android_winusb.inf

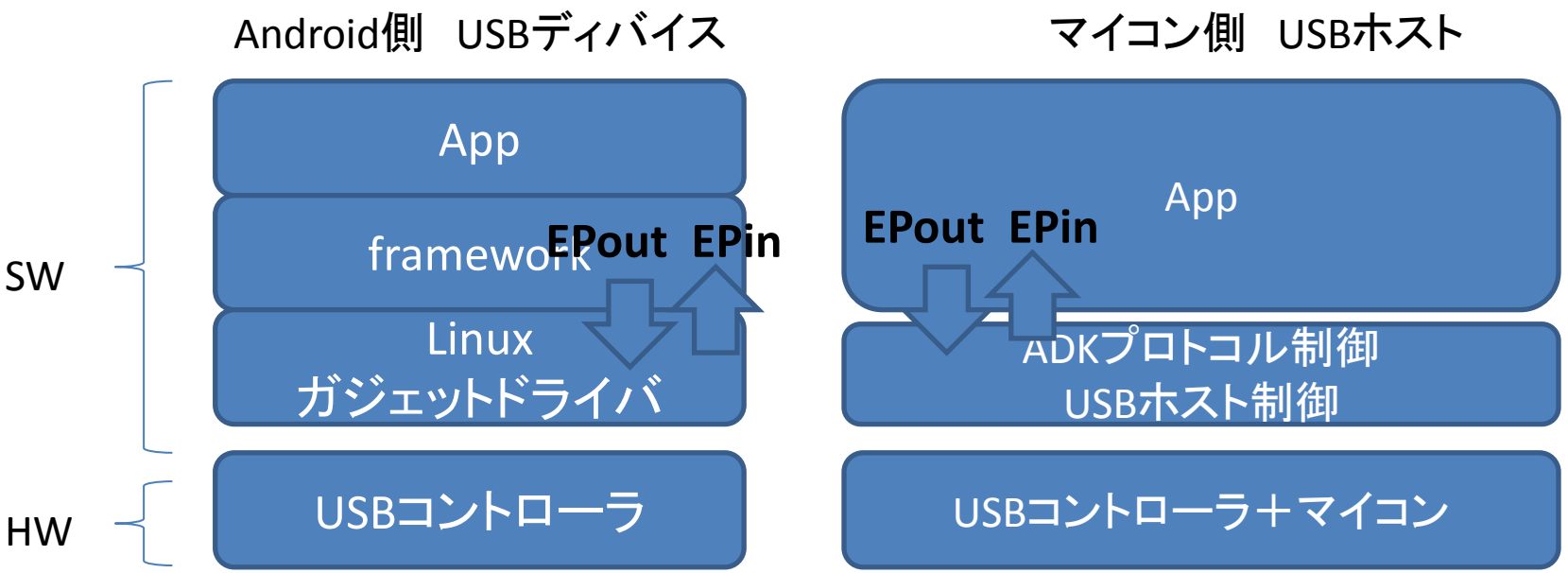
```
[Google.NTx86]
; HTC Dream
%SingleAdbInterface%        = USB_Install, USB¥VID_0BB4&PID_0C01
%CompositeAdbInterface%     = USB_Install, USB¥VID_0BB4&PID_0C02&MI_01
%SingleBootLoaderInterface% = USB_Install, USB¥VID_0BB4&PID_0FFF
; HTC Magic
%CompositeAdbInterface%     = USB_Install, USB¥VID_0BB4&PID_0C03&MI_01
;
```

Open Accessoryの構成

- 従来のAndroid側のUSBディバイスは
  - 特定の機能に限定
  - ユーザに非開放　アプリからは使えない
- 以前から、Beagleboard等の評価ボード乗りの間ではadbの拡張で自由な機能拡張が可能と認識されていた

Open Accessory modeとは何か

- adbなどのUSBガジェットの仕組みを流用
- 専用エンドポイントを提供
- フレームワークがアプリを起動し、エンドポイントを接続



Open Accessory protocol

1. マイコン側でUSBの接続を検出
2. Vid:Pidの0x18d1:0x2d00又は0x18d1:0x2d01かを確認できればaccessory mode 接続完了
3. ベンダ固有のIDが見えている場合があるので、’51’を送ってみる
4. 0以外が帰ってきたら、’52’で次の識別文字を送る
    1. manufacturer name: 0
    2. model name: 1
    3. description: 2
    4. version: 3
    5. URI: 4
    6. serial number: 5

    後で出てきます
5. 次に’53’を送り、2.へ

参考 : adbとaccessoryは兄弟

$ cd Eee_PAD_TF101/Kernel_V9_4_2_7/drivers/usb/gadget

$ diff -i f_accessory.c   f_adb.c

```
628c499
<       struct acc_dev                  *dev = func_to_dev(f);
---
>       struct adb_dev                  *dev = func_to_adb(f);
632,634c503,504
<       DBG(cdev, "acc_function_bind dev: %p¥n", dev);
<
<       dev->start_requested = 0;
---
>       dev->cdev = cdev;
>       DBG(cdev, "adb_function_bind dev: %p¥n", dev);
640c510
<       acc_interface_desc.bInterfaceNumber = id;
---
>       adb_interface_desc.bInterfaceNumber = id;
643,644c513,514
<       ret = create_bulk_endpoints(dev, &acc_fullspeed_in_desc,
<                                   &acc_fullspeed_out_desc);
---
>       ret = adb_create_bulk_endpoints(dev, &adb_fullspeed_in_desc,
>                                   &adb_fullspeed_out_desc);
650,653c520,523
<               acc_highspeed_in_desc.bEndpointAddress =
<                       acc_fullspeed_in_desc.bEndpointAddress;
<               acc_highspeed_out_desc.bEndpointAddress =
<                       acc_fullspeed_out_desc.bEndpointAddress;
---
>               adb_highspeed_in_desc.bEndpointAddress =
>                       adb_fullspeed_in_desc.bEndpointAddress;
>               adb_highspeed_out_desc.bEndpointAddress =
>                       adb_fullspeed_out_desc.bEndpointAddress;
663c533
< acc_function_unbind(struct usb_configuration *c, struct usb_function *f)
---
> adb_function_unbind(struct usb_configuration *c, struct usb_function *f)
665c535
<       struct acc_dev                  *dev = func_to_dev(f);
---
>       struct adb_dev                  *dev = func_to_adb(f);
```

PART4
マイコン側の仕組み

PICマイコン

- マイコンとしては老舗
- ライバルはAtmel AVR, ARM coretex-M
- PIC10,PIC12,PIC16,PIC18,PIC24,dsPIC
  - 独自アーキ
- PIC32シリーズ
  - MIPS社からライセンスしたMIPS M4K

マイコン（マイクロコントローラ）

- 小型、省電力のコントローラ
  - CPU
  - ROM(Flash),RAM内蔵 数K～数100Kbyte
  - 各種コントローラ内臓
    - GPIO
    - USART,I2C,AD変換,DA変換,PWM等
    - USB device, USB host
- PCでコンパイルして、ROM(Flash)に書き込んで利用

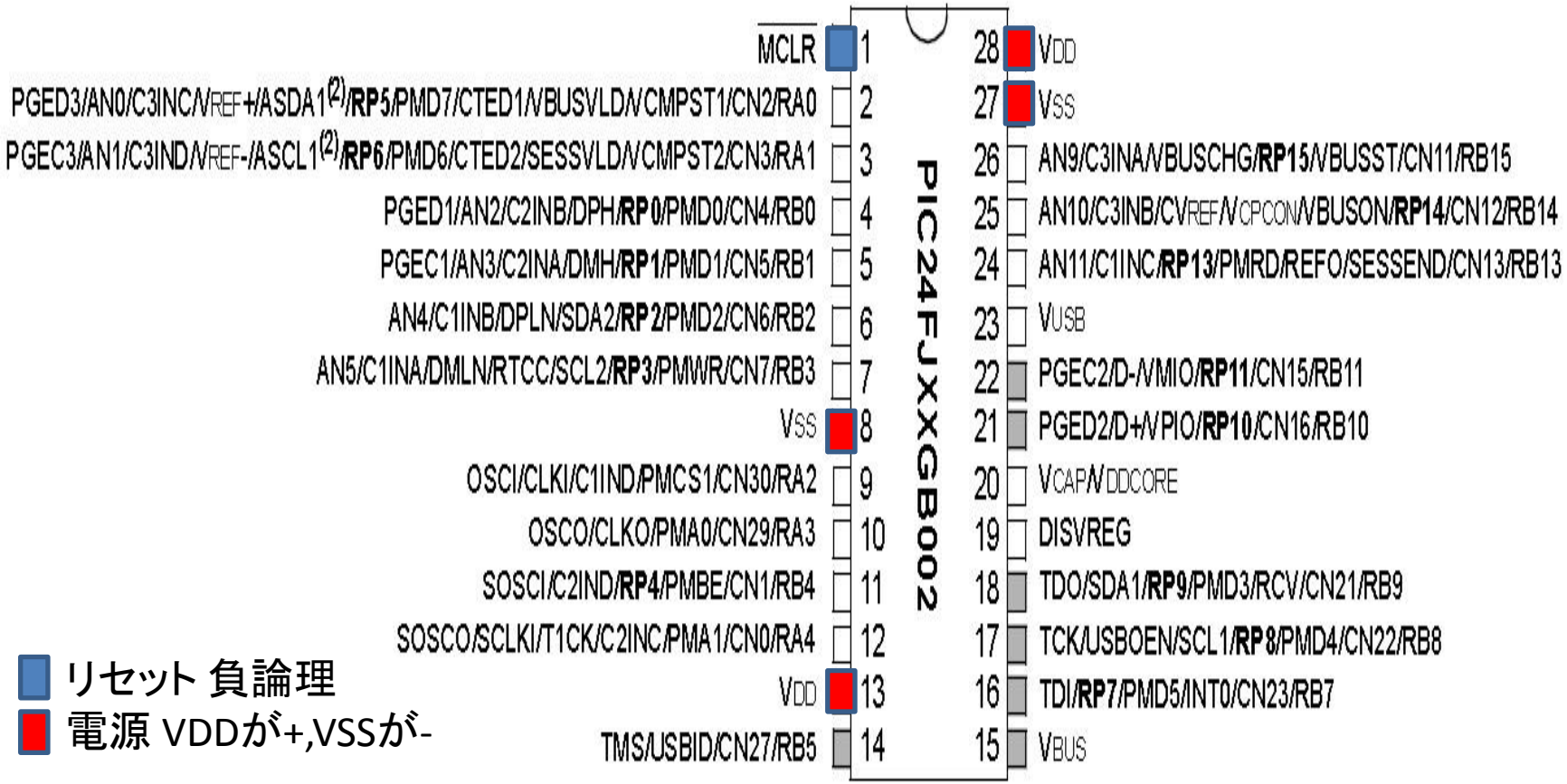
PIC24FJ64GB002の概要

- クロック最大32MHz/16MIPS
- ROM64Kbyte/RAM 8Kbyte
- 周辺回路
  - GPIO,ADC,PWM,UART,I2CSPI

参考までに、1979年発売のPC-8001

クロック4MHz/ROM 24Kbyte/RAM 16Kbyte/168,000円

PIC24FJ64GB002の電源・リセット



- リセット 負論理
- 電源 VDDが+,VSSが-

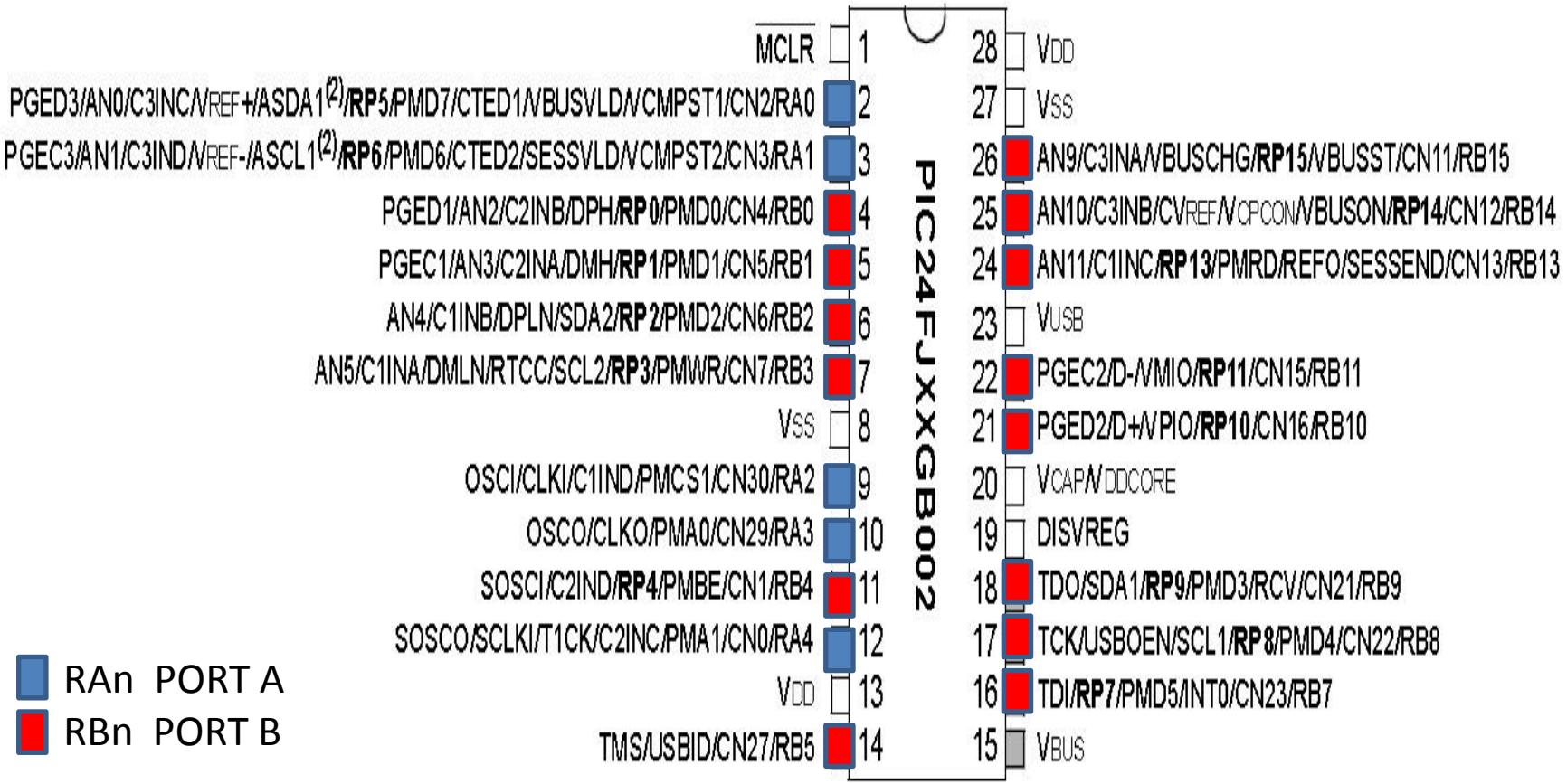
VDDとVSSに所定の電圧をかけてMCLRを軽くVDDに接続すればROMのプログラムが動作

GPIO ―汎用IO

- 出力
  - 特定のレジスタのビット操作で信号をON,OFF
- 入力
  - ピン(電極)の信号のON,OFFが特定のレジスタのビットを変更
- 1つのピンは入力・出力のどちらかを選択して利用する

レジスタのマップなどはPIC24FJ64GB004 Family Data Sheetから引用
http://ww1.microchip.com/downloads/en/DeviceDoc/39940d.pdf

PIC24FJ64GB002のGPIO



GPIO-レジスタマップ

- TRISA,TRISB　　入出力の向きを決めるレジスタ
  0:出力 1:入力
- PORTA,PORTB　　入力で使うレジスタ
  0:ON 1:OFF
- LATA,LATB　　出力で使うレジスタ
  0:ON 1:OFF

TABLE 4-12: PORTA REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10(1)	Bit 9(1)	Bit 8(1)	Bit 7(1)	Bit 6	Bit 5	Bit 4	Bit 3	Bit2	Bit 1	Bit 0	All Resets
TRISA	02C0	—	—	—	—	—	TRISA10	TRISA9	TRISA8	TRISA7	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	079F
PORTA	02C2	—	—	—	—	—	RA10	RA9	RA8	RA7	—	—	RA4	RA3	RA2	RA1	RA0	xxxx
LATA	02C4	—	—	—	—	—	LATA10	LATA9	LATA8	LATA7	—	—	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx
ODCA	02C6	—	—	—	—	—	ODA10	ODA9	ODA8	ODA7	—	—	ODA4	ODA3	ODA2	ODA1	ODA0	0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal. Reset values shown are for 44-pin devices.
Note 1: Bits are unimplemented in 28-pin devices; read as '0'.

TABLE 4-13: PORTB REGISTER MAP

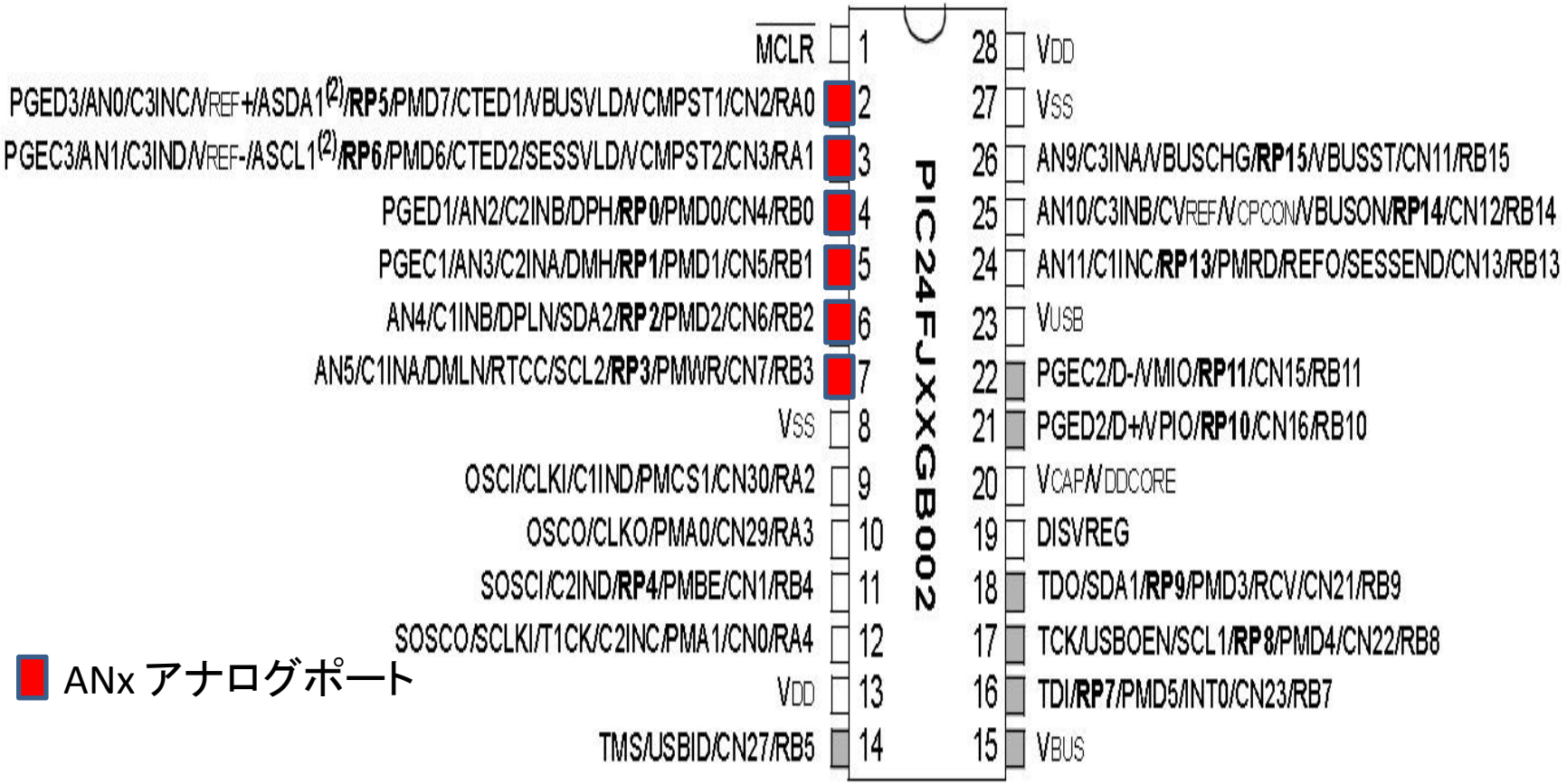
File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISB	02C8	TRISB15	TRISB14	TRISB13	—	TRISB11	TRISB10	TRISB9	TRISB8	TRISB7	—	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	EFBF
PORTB	02CA	RB15	RB14	RB13	—	RB11	RB10	RB9	RB8	RB7	—	RB5	RB4	RB3	RB2	RB1	RB0	xxxx
LATB	02CC	LATB15	LATB14	LATB13	—	LATB11	LATB10	LATB9	LATB8	LATB7	—	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx
ODCB	02CE	ODB15	ODB14	ODB13	—	ODB11	ODB10	ODB9	ODB8	ODB7	—	ODB5	ODB4	ODB3	ODB2	ODB1	ODB0	0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

ADC analog to digital converter

- 電圧で状態を出力する装置の値を読み取る
  - 温度計、湿度計、近接センサ、タッチパネル
- PIC24Fは10bit ADCなので
  - 0v～Nvまでを0～1023の解像度で読み取る
  - Nvは基準電圧で今回はVDDと同じ
- 今回のモジュールではセンサの代わりに可変抵抗を利用
  - 回転角、位置センサの読み取りとも言える

PIC24FJ64GB002のADC



ANx アナログポート

ADCのレジスタ

- 初期化
  - パラメタが多く複雑　レジスタ6個
  - 基準電圧=VDDの基本パターンを使いまわす
  - Firmwareソースのウォークスルーで紹介
- 読み取り
  - 時間を要する処理
    - 割り込み or ポーリング

開発環境

- MPLAB.X
  - 統合開発環境
  - 最近1.0に
  - NetBeansベース
    - MAC, Windows, Linuxの各プラットフォームで動作
- プログラマ
  - マイコン上のFlashにプログラムを転送
  - MPLAB.Xから書き込み
  - Pickit3

Pickit3 3900円
秋月　通販コード M-03608

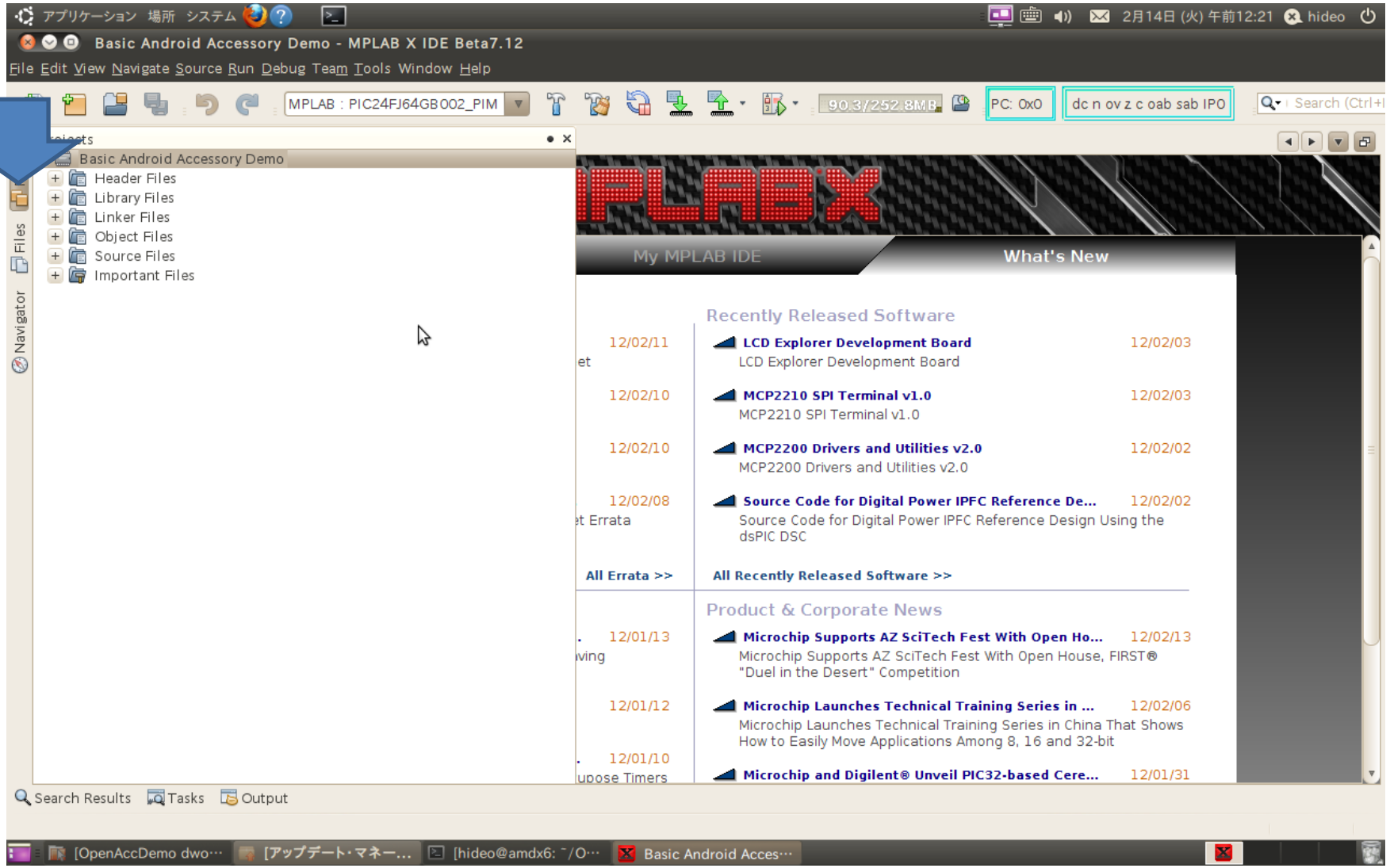
MPLABXの使い方

- プロジェクトの手順
- ビルドの手順
- プログラムの手順

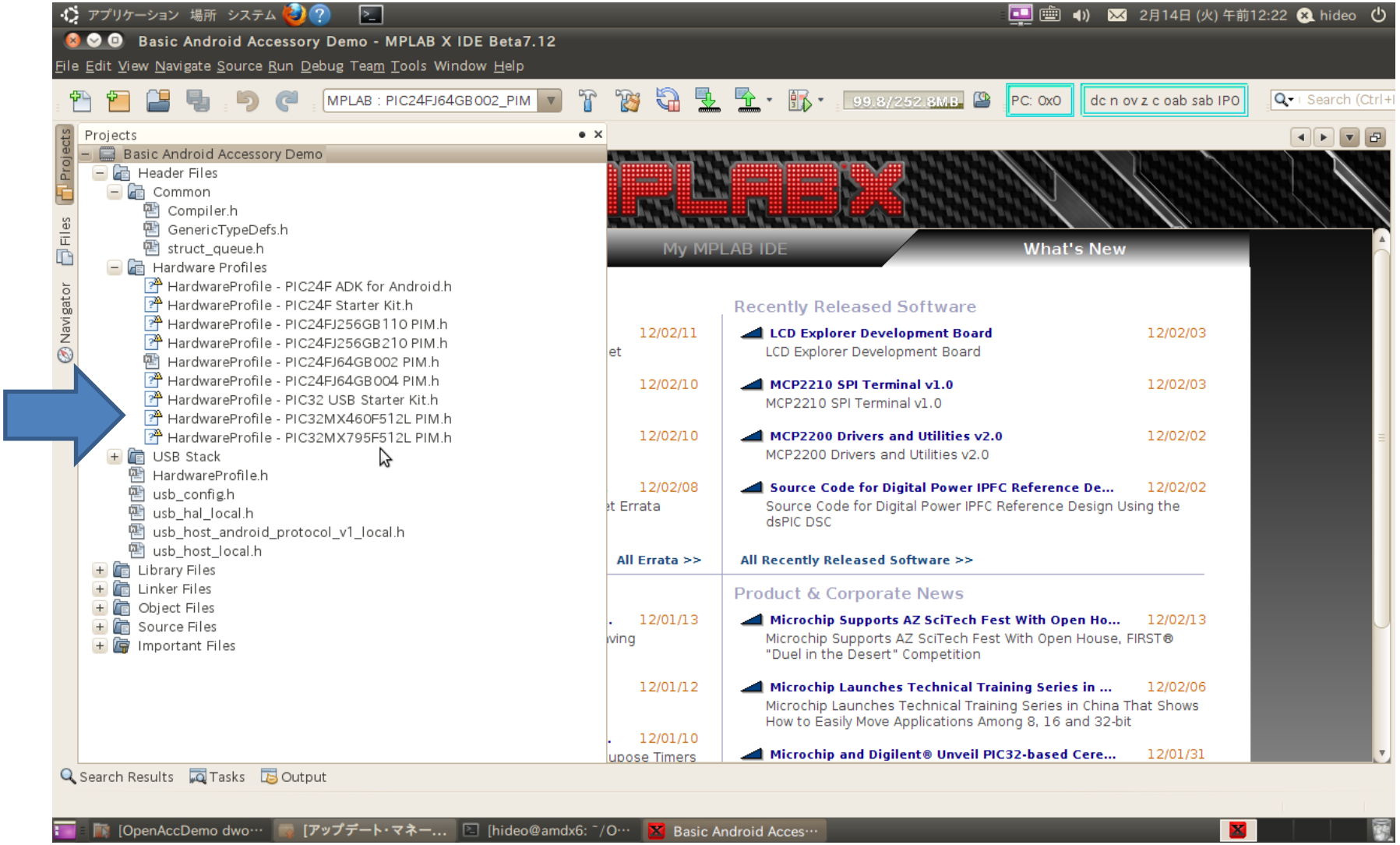
プロジェクトのOpen

- 左上メニューからFile -> Open Project….
- プロジェクトを選択
  - OpenAccessoryDemo/Firmware/MPXLABを選択

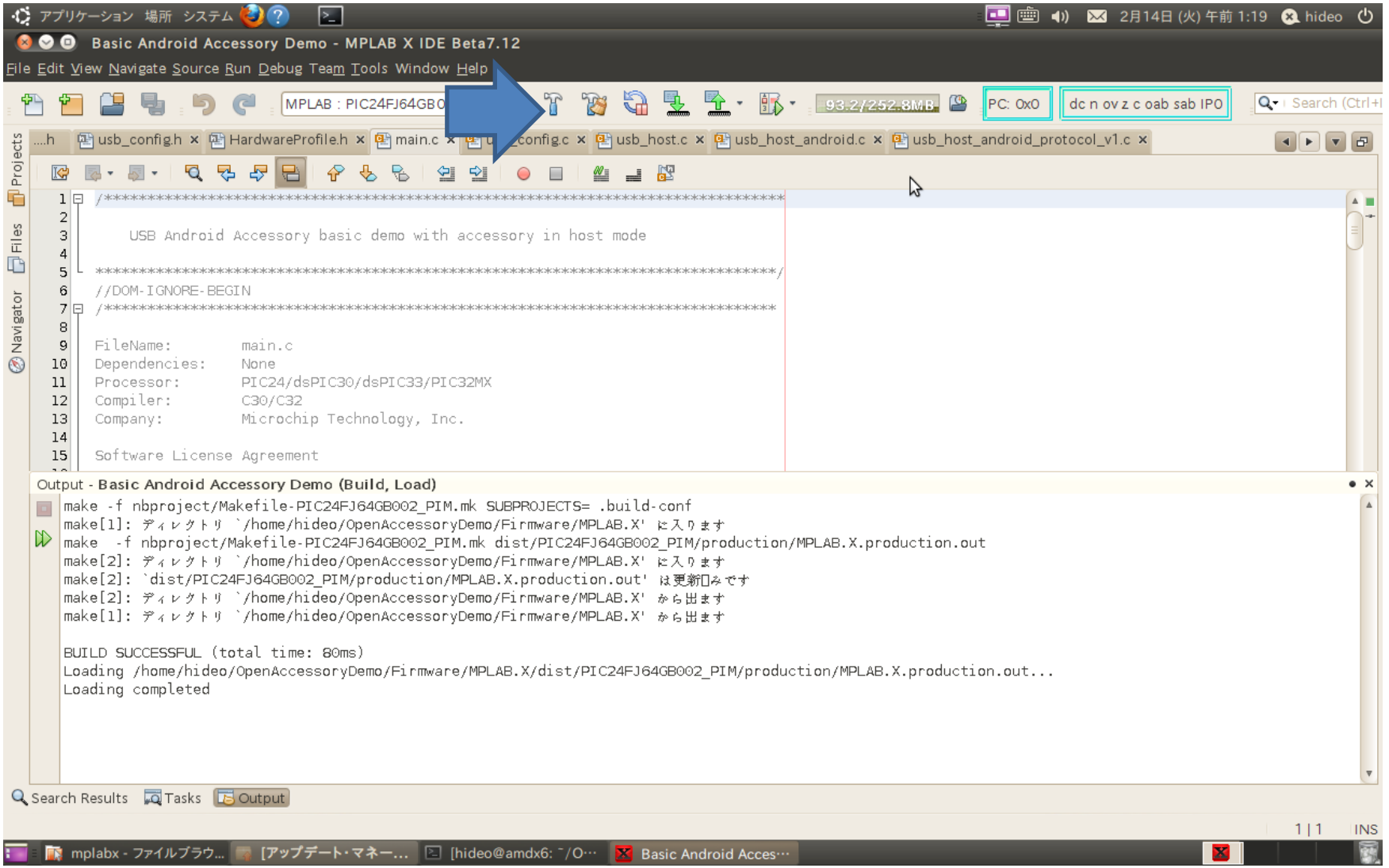
ファイルの閲覧


アプリケーション 場所 システム
2月14日 (火) 午前12:21 hideo
Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
PC: 0x0
Basic Android Accessory Demo
Header Files
Library Files
Linker Files
Object Files
Source Files
Important Files
Files
Navigator
My MPLAB IDE
What's New
Recently Released Software
LCD Explorer Development Board
LCD Explorer Development Board
MCP2210 SPI Terminal v1.0
MCP2210 SPI Terminal v1.0
MCP2200 Drivers and Utilities v2.0
MCP2200 Drivers and Utilities v2.0
Source Code for Digital Power IPFC Reference De...
Source Code for Digital Power IPFC Reference Design Using the dsPIC DSC
All Errata >>
All Recently Released Software >>
Product & Corporate News
Microchip Supports AZ SciTech Fest With Open Ho...
Microchip Supports AZ SciTech Fest With Open House, FIRST® "Duel in the Desert" Competition
Microchip Launches Technical Training Series in ...
Microchip Launches Technical Training Series in China That Shows How to Easily Move Applications Among 8, 16 and 32-bit
Microchip and Digilent® Unveil PIC32-based Cere...
Search Results
Tasks
Output
[OpenAccDemo dwo...
[アップデート・マネー...
[hideo@amdx6: ~/O...
Basic Android Acces...

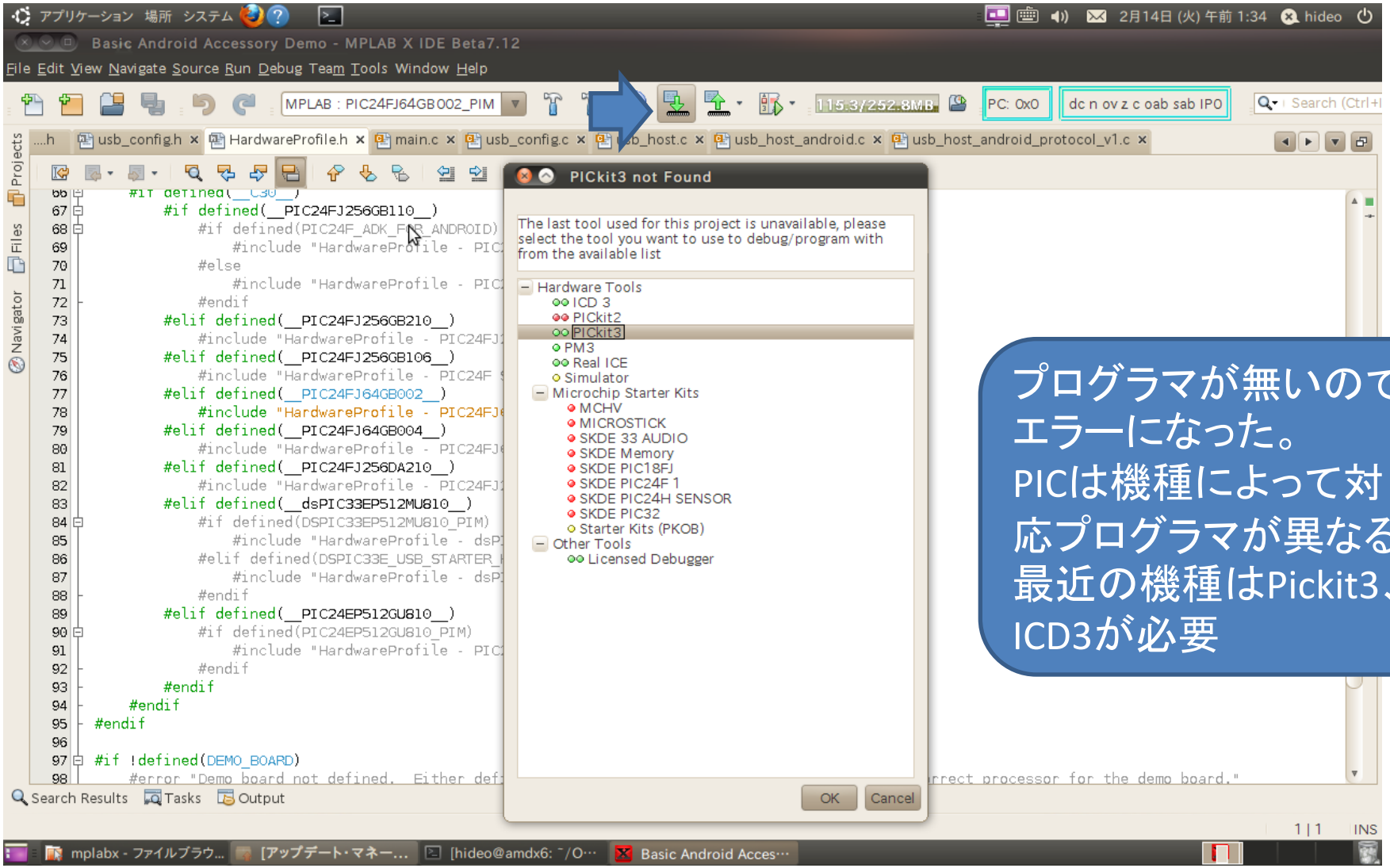
ファイルを選択


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
Projects
Basic Android Accessory Demo
Header Files
Common
Compiler.h
GenericTypeDefs.h
struct_queue.h
Hardware Profiles
HardwareProfile - PIC24F ADK for Android.h
HardwareProfile - PIC24F Starter Kit.h
HardwareProfile - PIC24FJ256GB110 PIM.h
HardwareProfile - PIC24FJ256GB210 PIM.h
HardwareProfile - PIC24FJ64GB002 PIM.h
HardwareProfile - PIC24FJ64GB004 PIM.h
HardwareProfile - PIC32 USB Starter Kit.h
HardwareProfile - PIC32MX460F512L PIM.h
HardwareProfile - PIC32MX795F512L PIM.h
USB Stack
HardwareProfile.h
usb_config.h
usb_hal_local.h
usb_host_android_protocol_v1_local.h
usb_host_local.h
Library Files
Linker Files
Object Files
Source Files
Important Files
My MPLAB IDE
What's New
Recently Released Software
LCD Explorer Development Board
MCP2210 SPI Terminal v1.0
MCP2200 Drivers and Utilities v2.0
Source Code for Digital Power IPFC Reference De...
All Errata >>
All Recently Released Software >>
Product & Corporate News
Microchip Supports AZ SciTech Fest With Open Ho...
Microchip Launches Technical Training Series in ...
Microchip and Digilent® Unveil PIC32-based Cere...
Search Results
Tasks
Output

ビルド


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
usb_config.h
HardwareProfile.h
main.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
USB Android Accessory basic demo with accessory in host mode
FileName: main.c
Dependencies: None
Processor: PIC24/dsPIC30/dsPIC33/PIC32MX
Compiler: C30/C32
Company: Microchip Technology, Inc.
Software License Agreement
Output - Basic Android Accessory Demo (Build, Load)
make -f nbproject/Makefile-PIC24FJ64GB002_PIM.mk SUBPROJECTS= .build-conf
make[1]: ディレクトリ `/home/hideo/OpenAccessoryDemo/Firmware/MPLAB.X' に入ります
make -f nbproject/Makefile-PIC24FJ64GB002_PIM.mk dist/PIC24FJ64GB002_PIM/production/MPLAB.X.production.out
make[2]: ディレクトリ `/home/hideo/OpenAccessoryDemo/Firmware/MPLAB.X' に入ります
make[2]: `dist/PIC24FJ64GB002_PIM/production/MPLAB.X.production.out' は更新Ｏです
make[2]: ディレクトリ `/home/hideo/OpenAccessoryDemo/Firmware/MPLAB.X' から出ます
make[1]: ディレクトリ `/home/hideo/OpenAccessoryDemo/Firmware/MPLAB.X' から出ます
BUILD SUCCESSFUL (total time: 80ms)
Loading /home/hideo/OpenAccessoryDemo/Firmware/MPLAB.X/dist/PIC24FJ64GB002_PIM/production/MPLAB.X.production.out...
Loading completed
Search Results
Tasks
Output

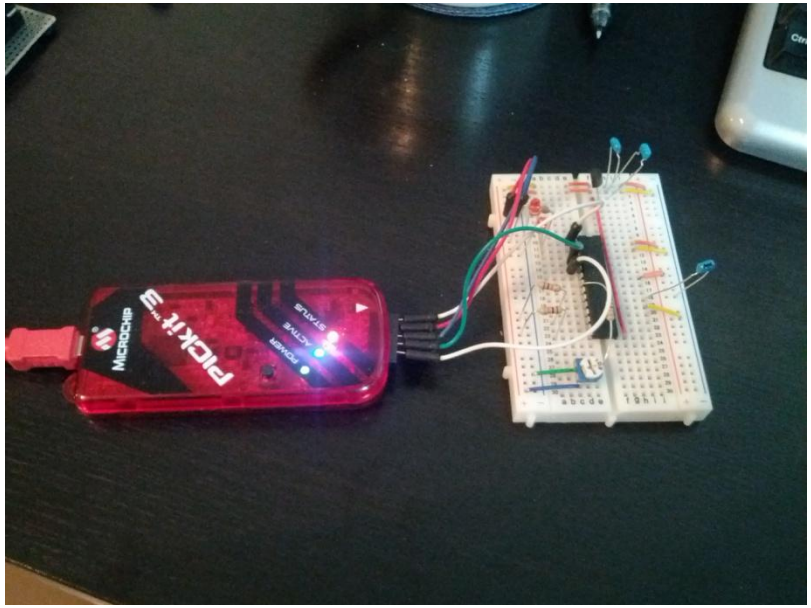
プログラム（Flashへの書き込み）



プログラマが無いのでエラーになった。
PICは機種によって対応プログラマが異なる。
最近の機種はPickit3、ICD3が必要

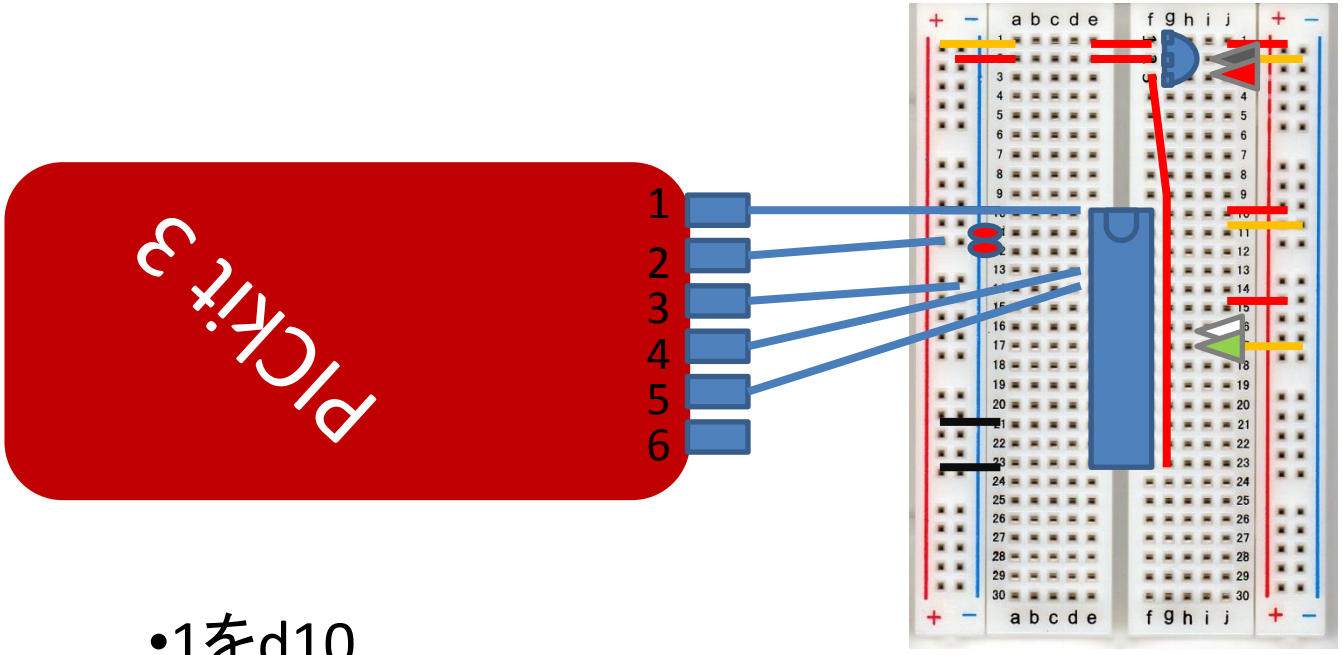
プログラマの接続

方式１）専用ソケットを用意



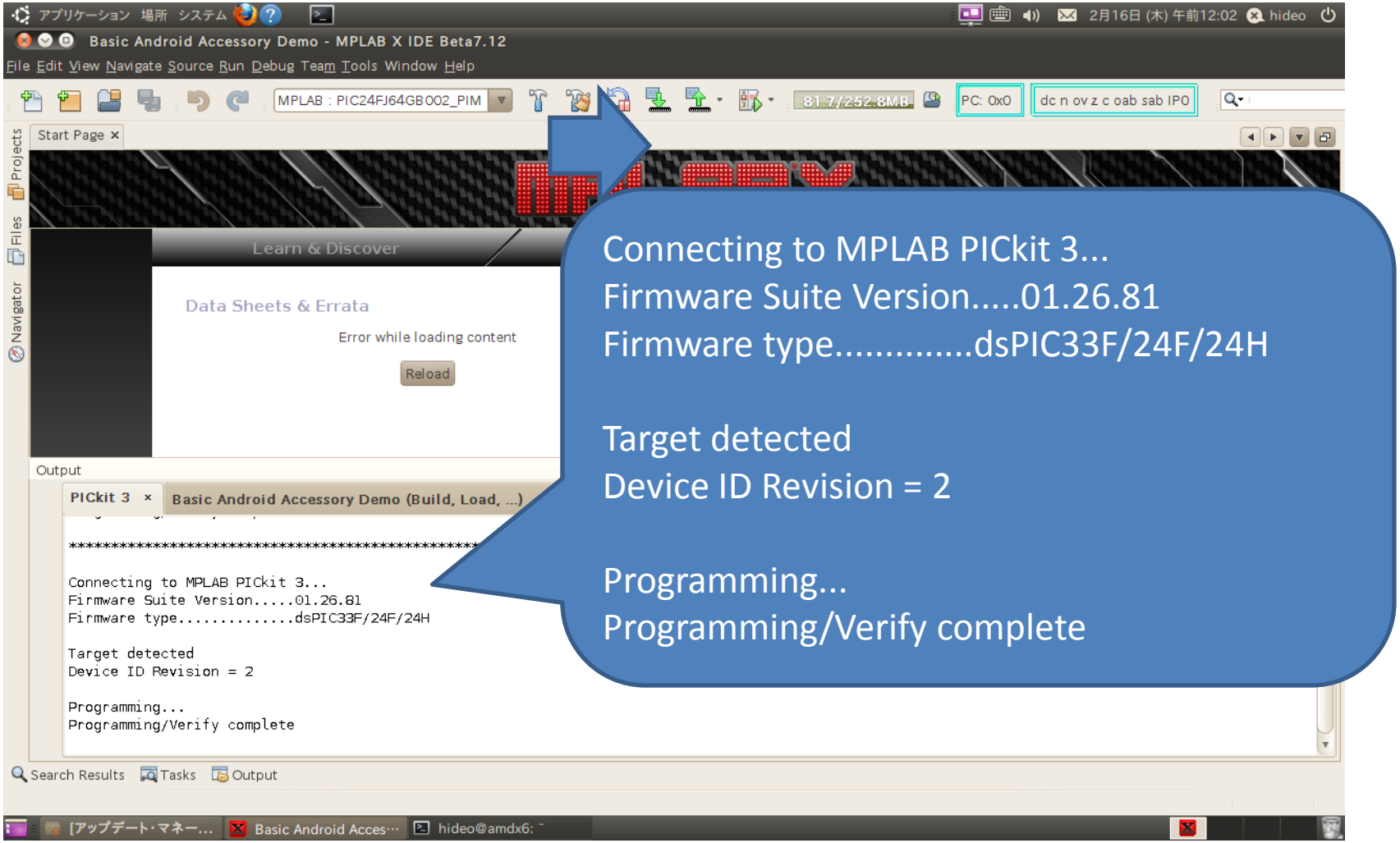
方式２）ICSP
In Circuit Serial Programming
基板上にPICをさしたまま、Flashへ書きこむ

プログラマの配線



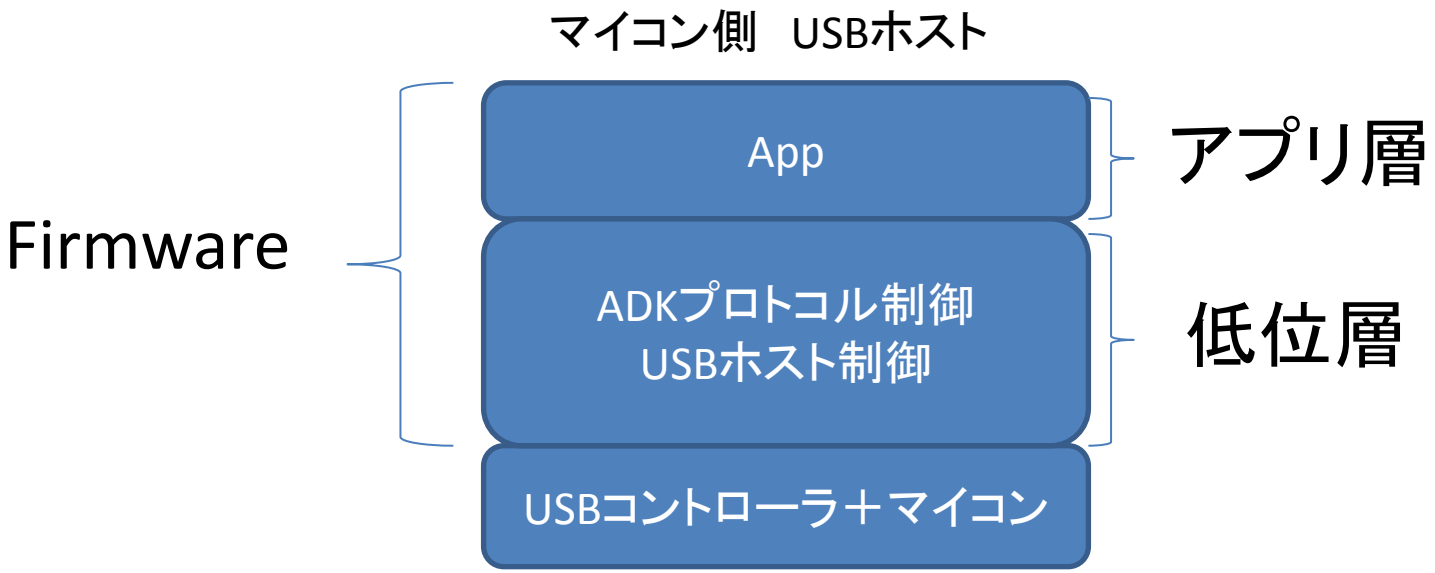
- 1をd10
- 2を＋
- 3を－
- 4をd13
- 5をd14

プログラム

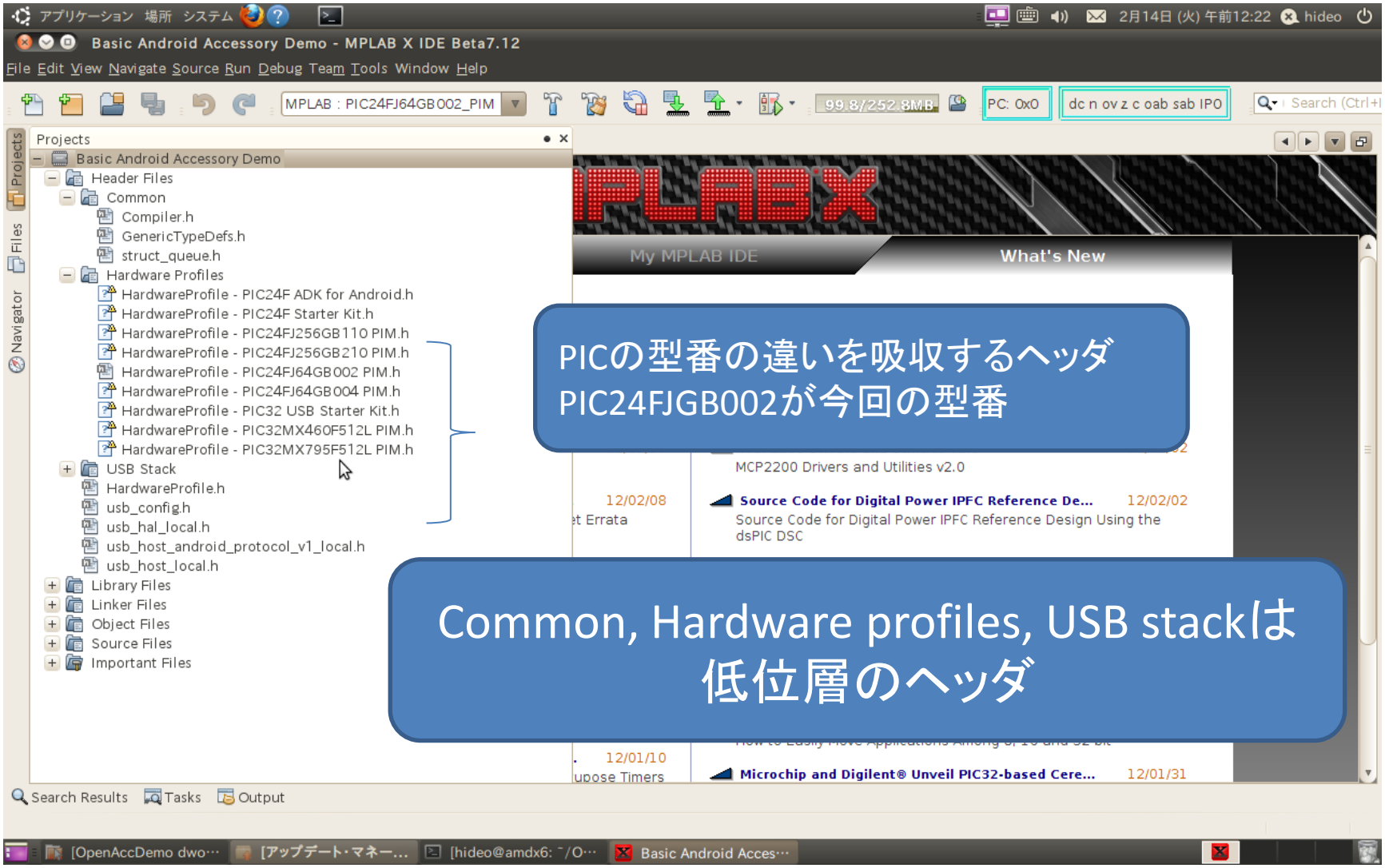

Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
Start Page
Learn & Discover
Data Sheets & Errata
Error while loading content
Reload
Output
PICkit 3
Basic Android Accessory Demo (Build, Load, ...)
Connecting to MPLAB PICkit 3...
Firmware Suite Version.....01.26.81
Firmware type.....dsPIC33F/24F/24H
Target detected
Device ID Revision = 2
Programming...
Programming/Verify complete
Search Results
Tasks
Output
Connecting to MPLAB PICkit 3...
Firmware Suite Version.....01.26.81
Firmware type.....dsPIC33F/24F/24H
Target detected
Device ID Revision = 2
Programming...
Programming/Verify complete

Firmwareのウォークスルー

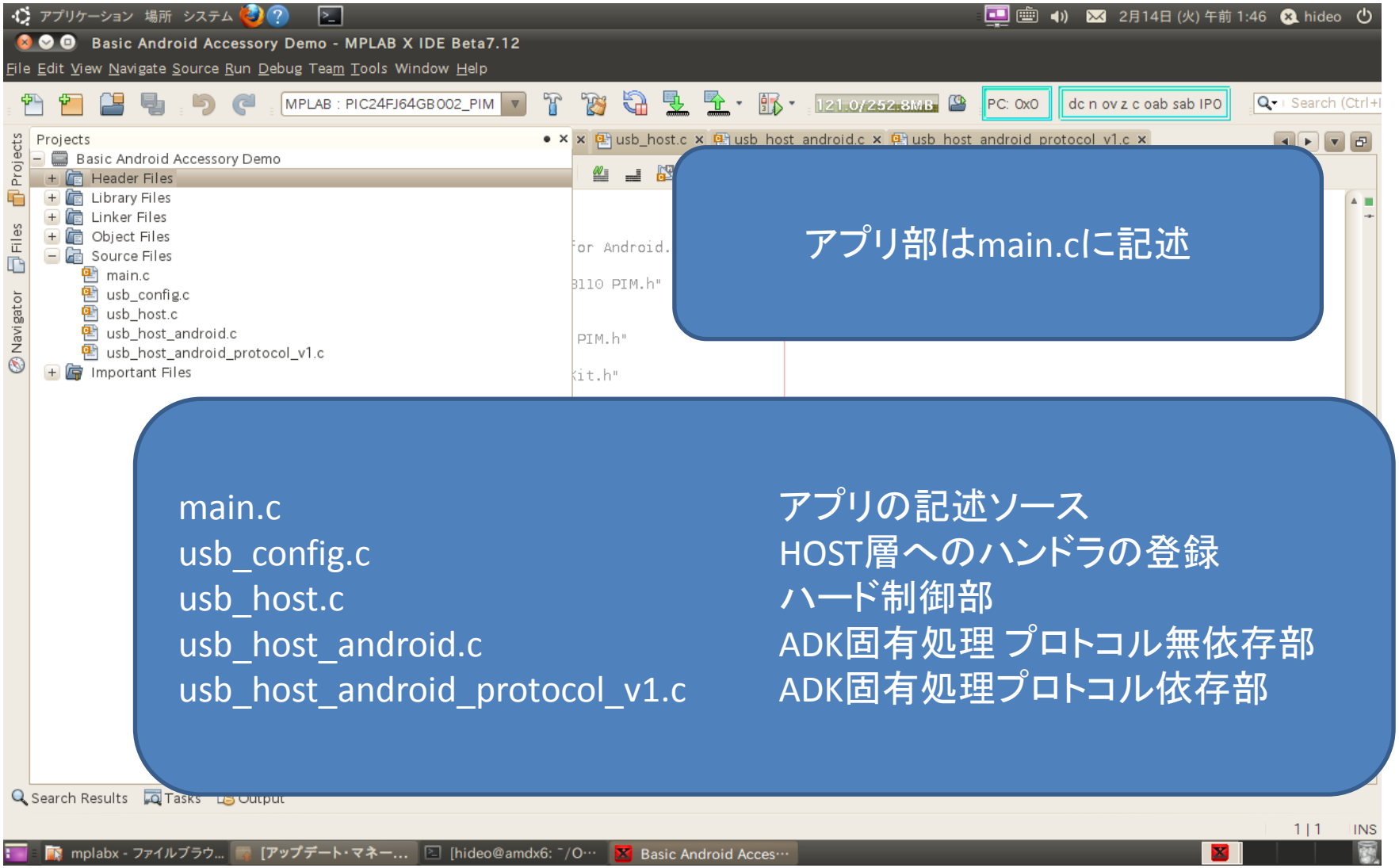
- Firmwareには、USBの低位層からアプリ層までのプログラムがソースで配置
- アプリ層のファイルは



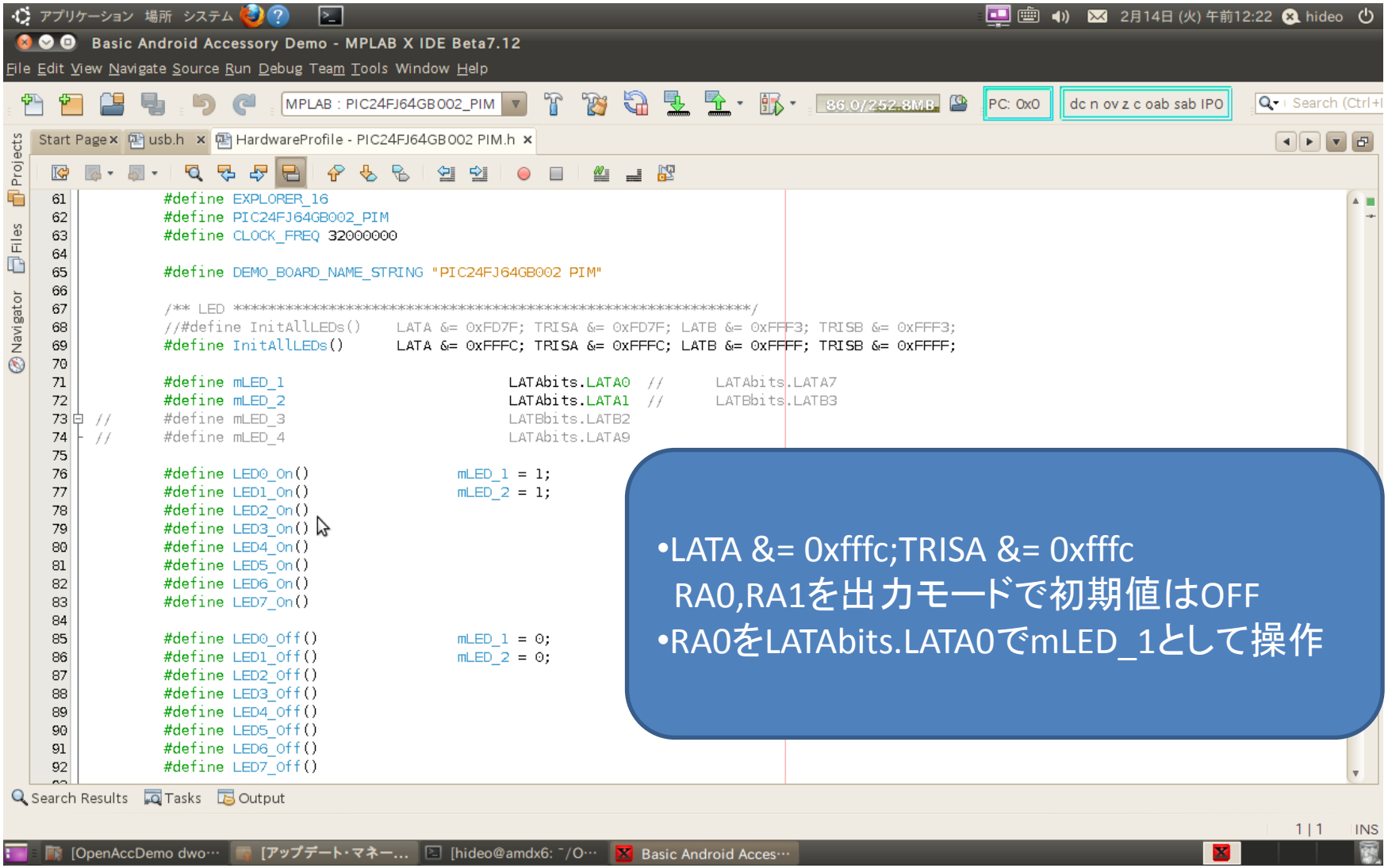
低位層のヘッダ


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
Projects
Basic Android Accessory Demo
Header Files
Common
Compiler.h
GenericTypeDefs.h
struct_queue.h
Hardware Profiles
HardwareProfile - PIC24F ADK for Android.h
HardwareProfile - PIC24F Starter Kit.h
HardwareProfile - PIC24FJ256GB110 PIM.h
HardwareProfile - PIC24FJ256GB210 PIM.h
HardwareProfile - PIC24FJ64GB002 PIM.h
HardwareProfile - PIC24FJ64GB004 PIM.h
HardwareProfile - PIC32 USB Starter Kit.h
HardwareProfile - PIC32MX460F512L PIM.h
HardwareProfile - PIC32MX795F512L PIM.h
USB Stack
HardwareProfile.h
usb_config.h
usb_hal_local.h
usb_host_android_protocol_v1_local.h
usb_host_local.h
Library Files
Linker Files
Object Files
Source Files
Important Files
My MPLAB IDE
What's New
MCP2200 Drivers and Utilities v2.0
Source Code for Digital Power IPFC Reference De...
Source Code for Digital Power IPFC Reference Design Using the dsPIC DSC
Microchip and Digilent® Unveil PIC32-based Cere...
Search Results
Tasks
Output
PICの型番の違いを吸収するヘッダ
PIC24FJGB002が今回の型番
Common, Hardware profiles, USB stackは
低位層のヘッダ

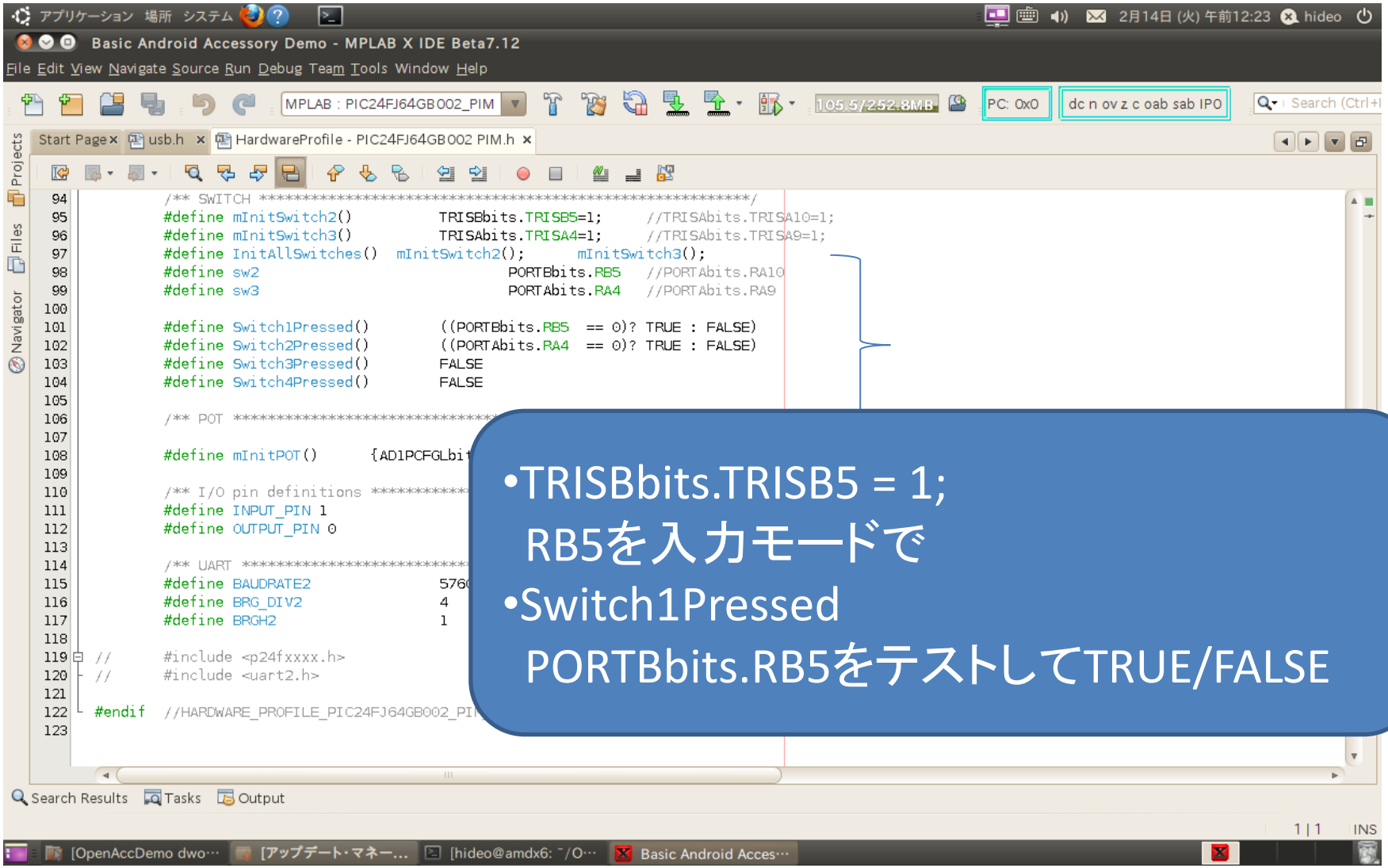
ソースファイル


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
Projects
Basic Android Accessory Demo
Header Files
Library Files
Linker Files
Object Files
Source Files
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
Important Files
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
Search Results
Tasks
Output
アプリ部はmain.cに記述
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
アプリの記述ソース
HOST層へのハンドラの登録
ハード制御部
ADK固有処理 プロトコル無依存部
ADK固有処理プロトコル依存部

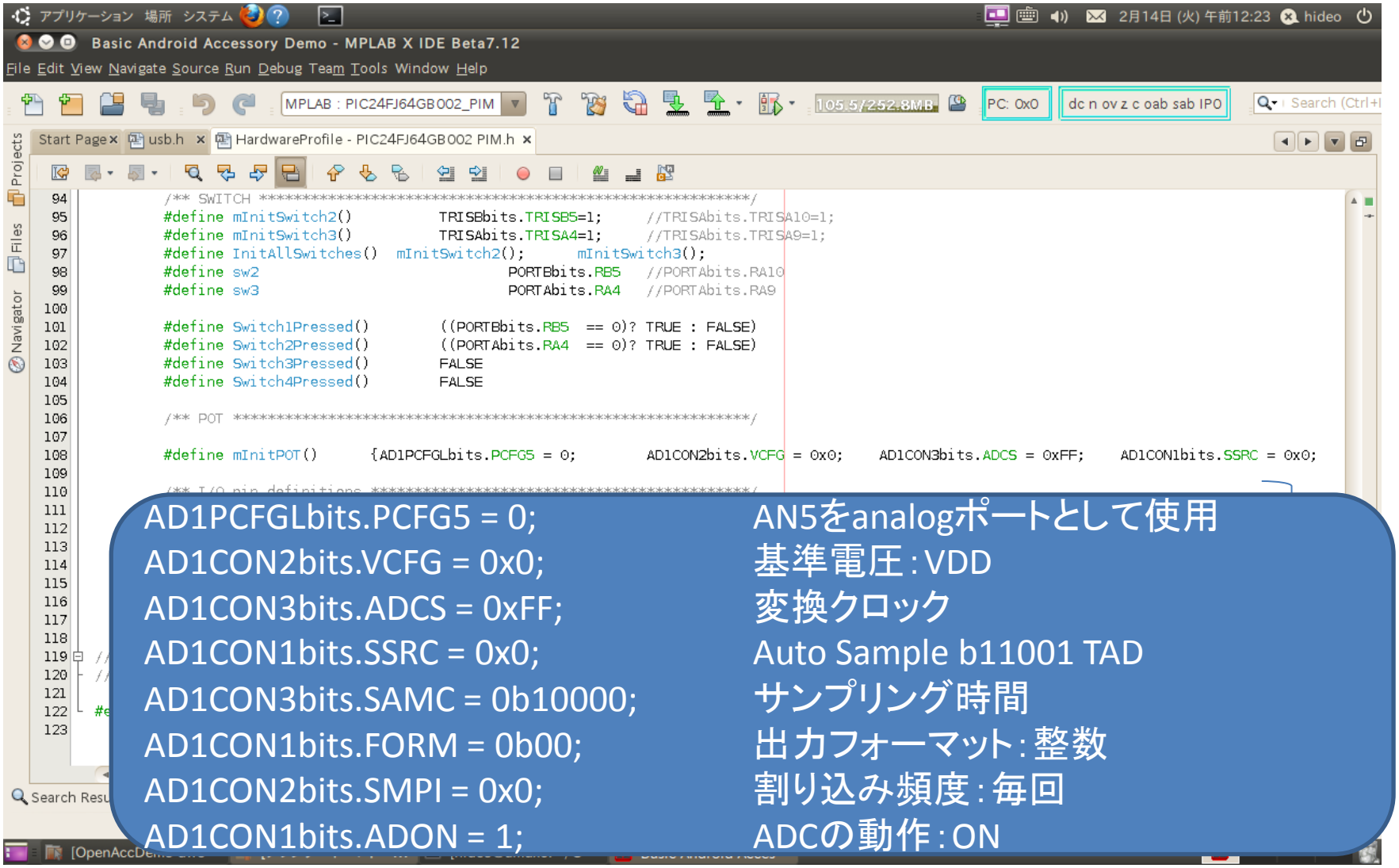
GPIOポートの出力設定



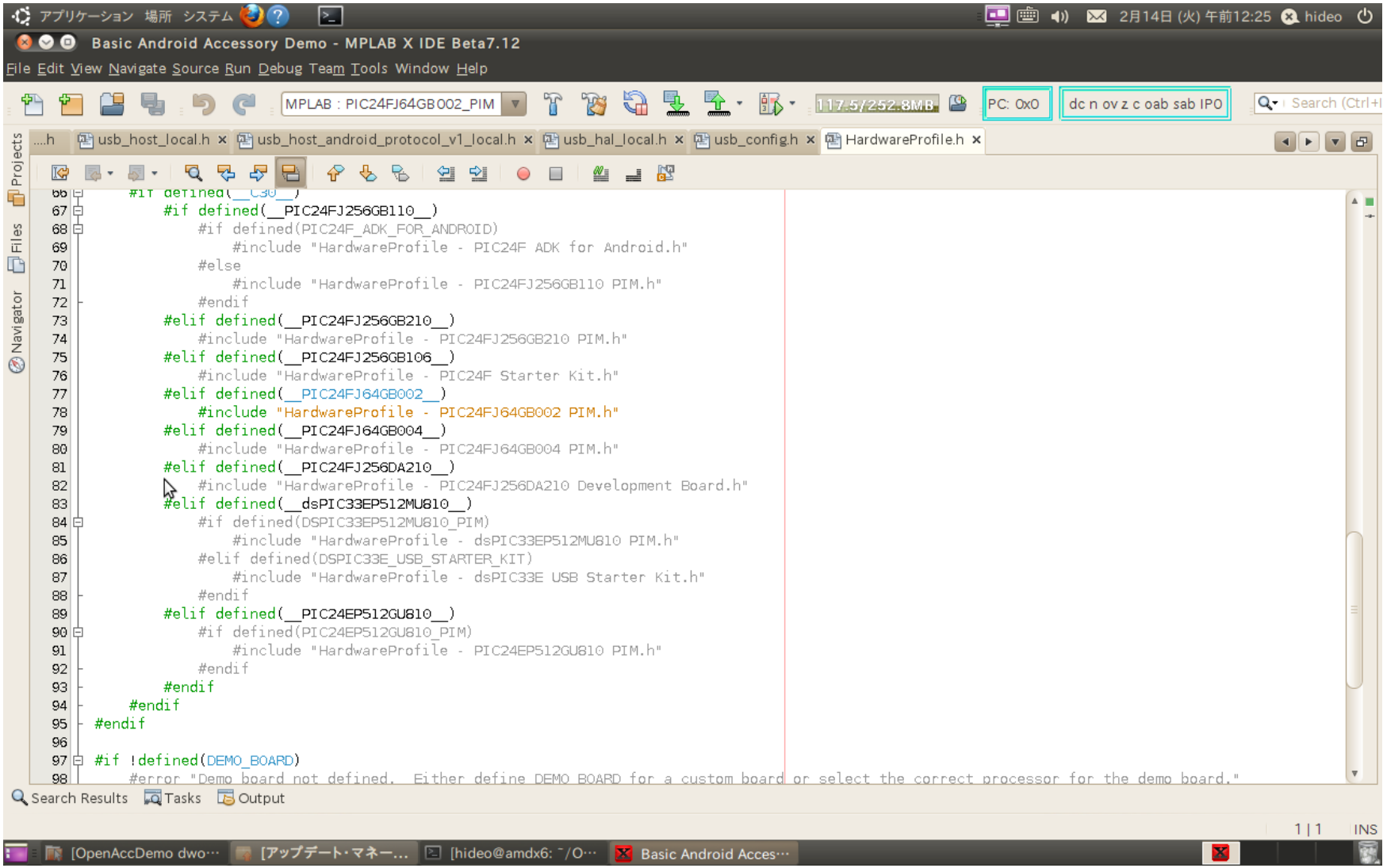
GPIOの入力設定


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
Start Page
usb.h
HardwareProfile - PIC24FJ64GB002 PIM.h
94 /** SWITCH ****/
95 #define mInitSwitch2() TRISBbits.TRISB5=1; //TRISAbits.TRISA10=1;
96 #define mInitSwitch3() TRISAbits.TRISA4=1; //TRISAbits.TRISA9=1;
97 #define InitAllSwitches() mInitSwitch2(); mInitSwitch3();
98 #define sw2 PORTBbits.RB5 //PORTAbits.RA10
99 #define sw3 PORTAbits.RA4 //PORTAbits.RA9
100
101 #define Switch1Pressed() ((PORTBbits.RB5 == 0)? TRUE : FALSE)
102 #define Switch2Pressed() ((PORTAbits.RA4 == 0)? TRUE : FALSE)
103 #define Switch3Pressed() FALSE
104 #define Switch4Pressed() FALSE
105
106 /** POT ****
107
108 #define mInitPOT() {AD1PCFGLbi
109
110 /** I/O pin definitions ****
111 #define INPUT_PIN 1
112 #define OUTPUT_PIN 0
113
114 /** UART ****
115 #define BAUDRATE2 576
116 #define BRG_DIV2 4
117 #define BRGH2 1
118
119 // #include <p24fxxxx.h>
120 // #include <uart2.h>
121
122 #endif //HARDWARE_PROFILE_PIC24FJ64GB002_PI
123
Search Results
Tasks
Output
•TRISBbits.TRISB5 = 1;
RB5を入力モードで
•Switch1Pressed
PORTBbits.RB5をテストしてTRUE/FALSE

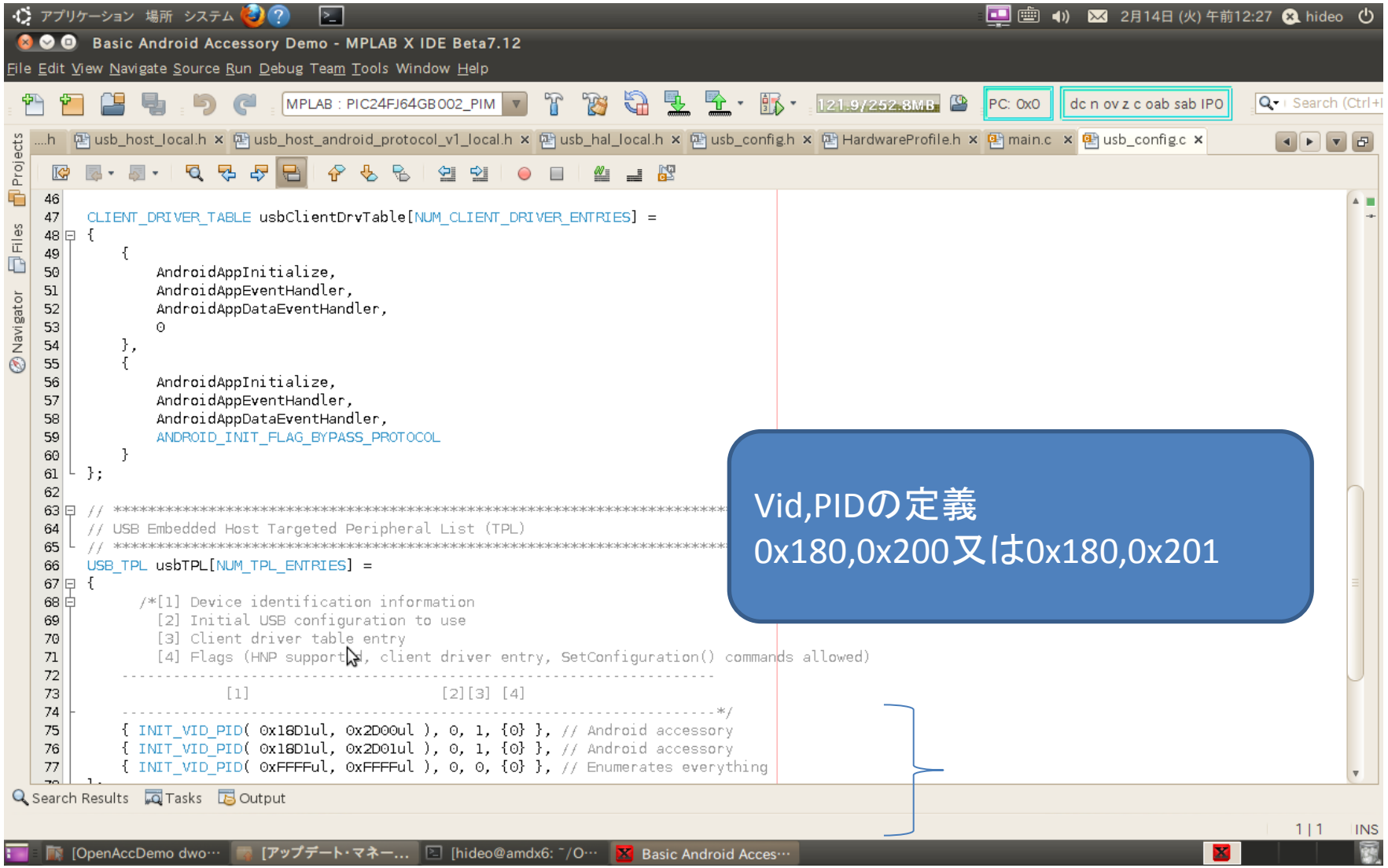
ADCの設定


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
Start Page
usb.h
HardwareProfile - PIC24FJ64GB002 PIM.h
#define mInitSwitch2() TRISBbits.TRISB5=1; //TRISAbits.TRISA10=1;
#define mInitSwitch3() TRISAbits.TRISA4=1; //TRISAbits.TRISA9=1;
#define InitAllSwitches() mInitSwitch2(); mInitSwitch3();
#define sw2 PORTBbits.RB5 //PORTAbits.RA10
#define sw3 PORTAbits.RA4 //PORTAbits.RA9
#define Switch1Pressed() ((PORTBbits.RB5 == 0)? TRUE : FALSE)
#define Switch2Pressed() ((PORTAbits.RA4 == 0)? TRUE : FALSE)
#define Switch3Pressed() FALSE
#define Switch4Pressed() FALSE
#define mInitPOT() {AD1PCFGLbits.PCFG5 = 0; AD1CON2bits.VCFG = 0x0; AD1CON3bits.ADCS = 0xFF; AD1CON1bits.SSRC = 0x0;
AD1PCFGLbits.PCFG5 = 0; AN5をanalogポートとして使用
AD1CON2bits.VCFG = 0x0; 基準電圧：VDD
AD1CON3bits.ADCS = 0xFF; 変換クロック
AD1CON1bits.SSRC = 0x0; Auto Sample b11001 TAD
AD1CON3bits.SAMC = 0b10000; サンプリング時間
AD1CON1bits.FORM = 0b00; 出力フォーマット：整数
AD1CON2bits.SMPI = 0x0; 割り込み頻度：毎回
AD1CON1bits.ADON = 1; ADCの動作：ON

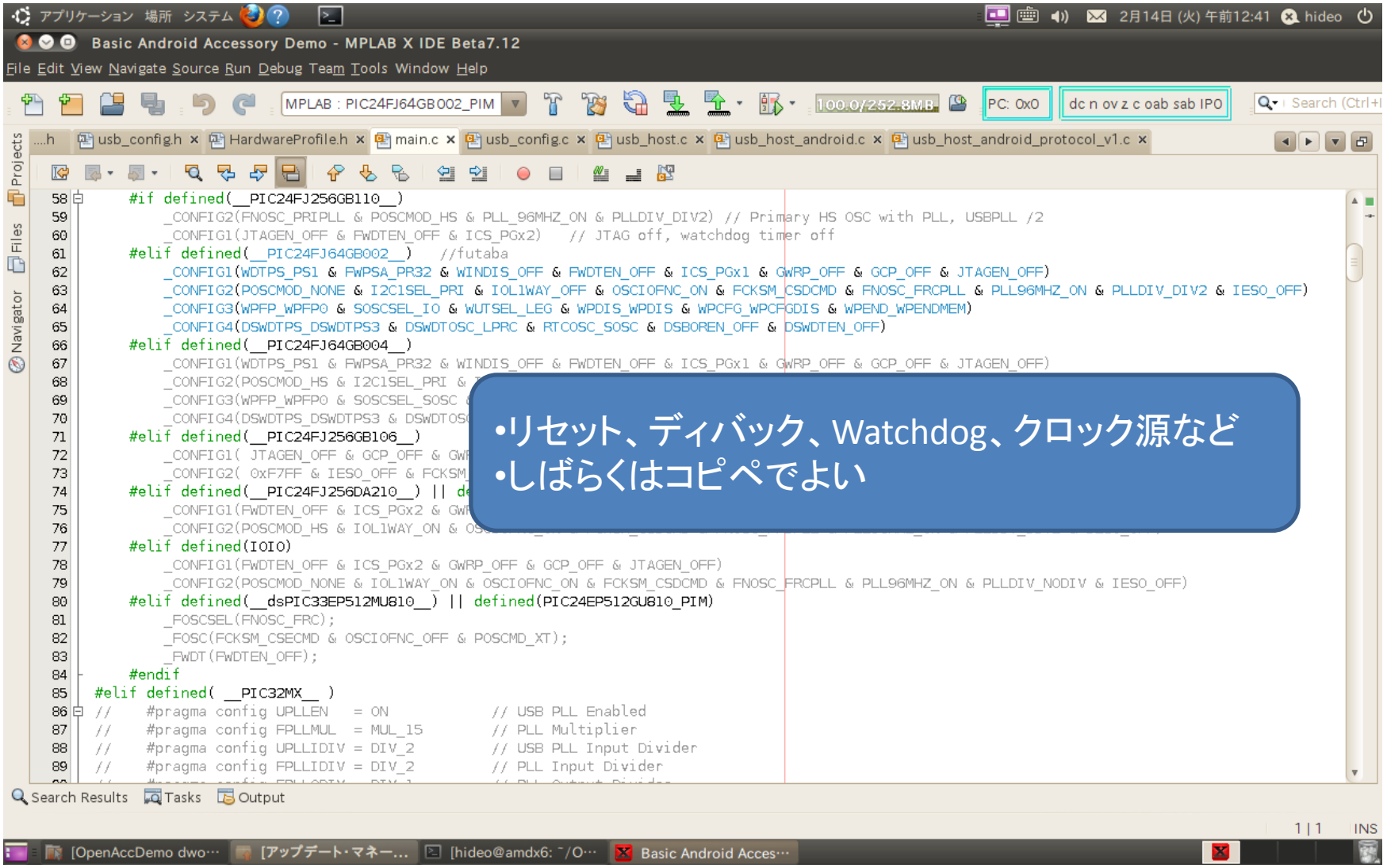
型番依存部の切り替え


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
usb_host_local.h
usb_host_android_protocol_v1_local.h
usb_hal_local.h
usb_config.h
HardwareProfile.h
#if defined(__PIC24FJ256GB110__)
#if defined(PIC24F_ADK_FOR_ANDROID)
#include "HardwareProfile - PIC24F ADK for Android.h"
#else
#include "HardwareProfile - PIC24FJ256GB110 PIM.h"
#endif
#elif defined(__PIC24FJ256GB210__)
#include "HardwareProfile - PIC24FJ256GB210 PIM.h"
#elif defined(__PIC24FJ256GB106__)
#include "HardwareProfile - PIC24F Starter Kit.h"
#elif defined(__PIC24FJ64GB002__)
#include "HardwareProfile - PIC24FJ64GB002 PIM.h"
#elif defined(__PIC24FJ64GB004__)
#include "HardwareProfile - PIC24FJ64GB004 PIM.h"
#elif defined(__PIC24FJ256DA210__)
#include "HardwareProfile - PIC24FJ256DA210 Development Board.h"
#elif defined(__dsPIC33EP512MU810__)
#if defined(DSPIC33EP512MU810_PIM)
#include "HardwareProfile - dsPIC33EP512MU810 PIM.h"
#elif defined(DSPIC33E_USB_STARTER_KIT)
#include "HardwareProfile - dsPIC33E USB Starter Kit.h"
#endif
#elif defined(__PIC24EP512GU810__)
#if defined(PIC24EP512GU810_PIM)
#include "HardwareProfile - PIC24EP512GU810 PIM.h"
#endif
#endif
#endif
#if !defined(DEMO_BOARD)
#error "Demo board not defined. Either define DEMO_BOARD for a custom board or select the correct processor for the demo board."
Search Results
Tasks
Output

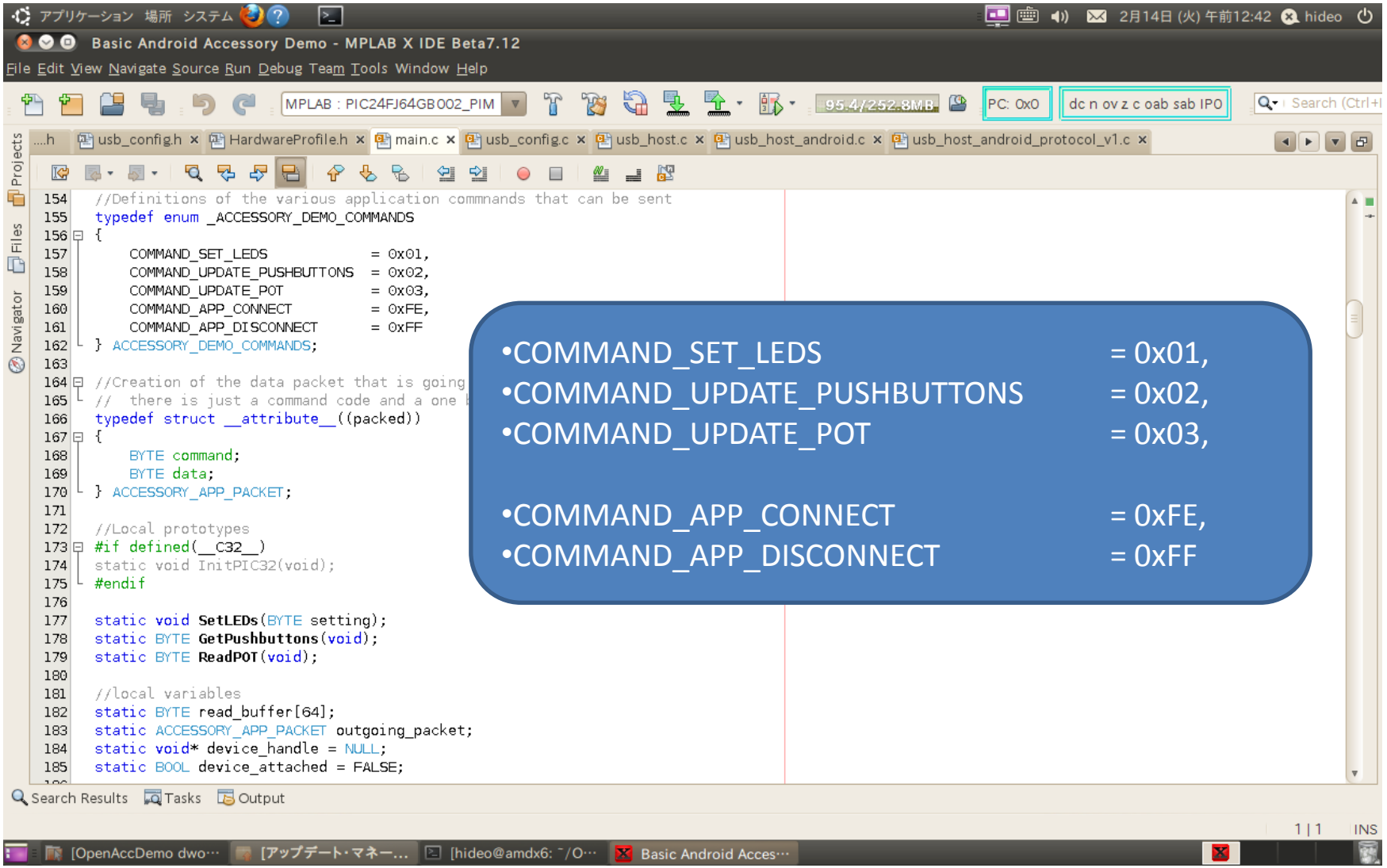
処理対象のVid,Pidの定義



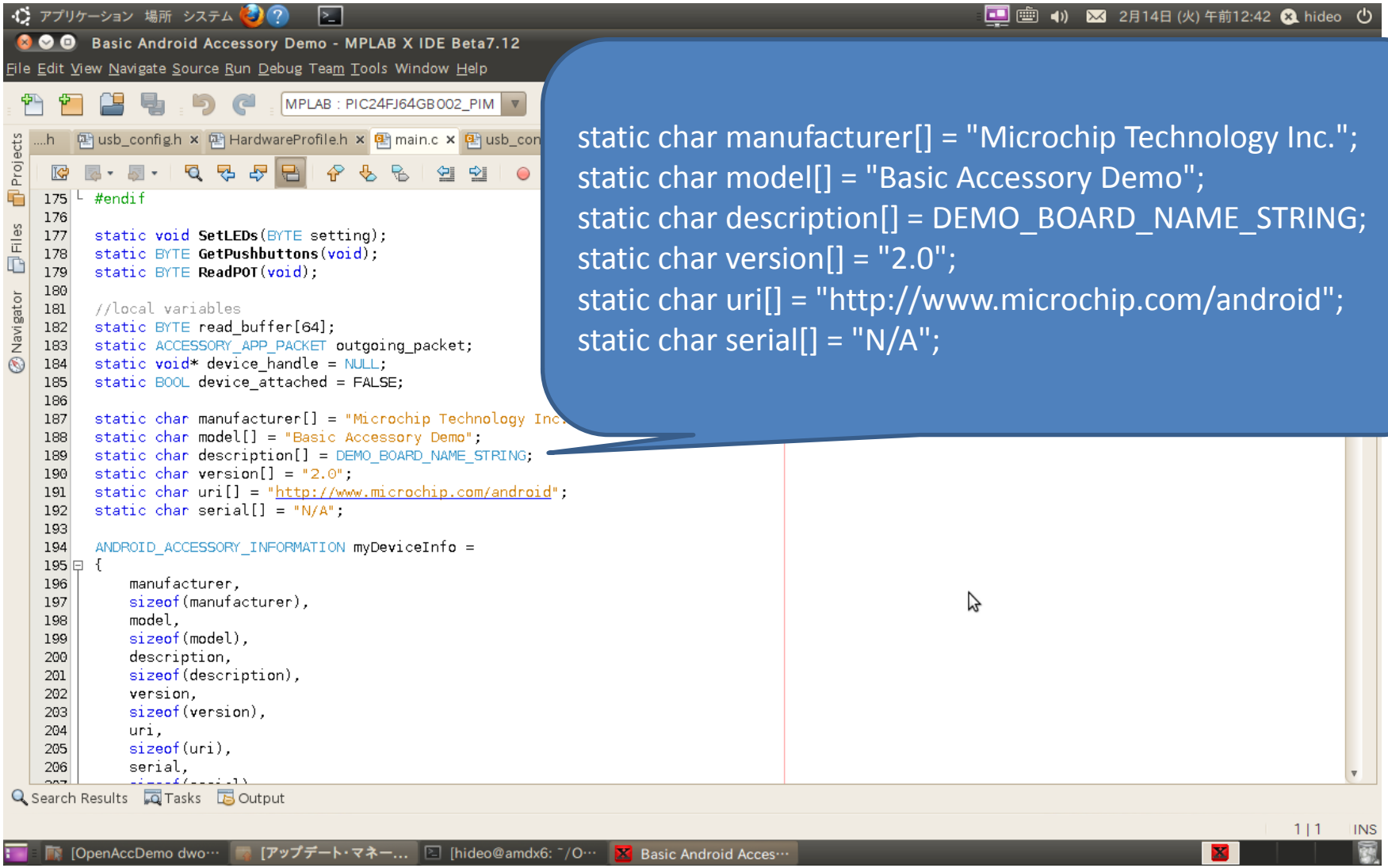
PICのファンクション指定



Appとのプロトコル定義

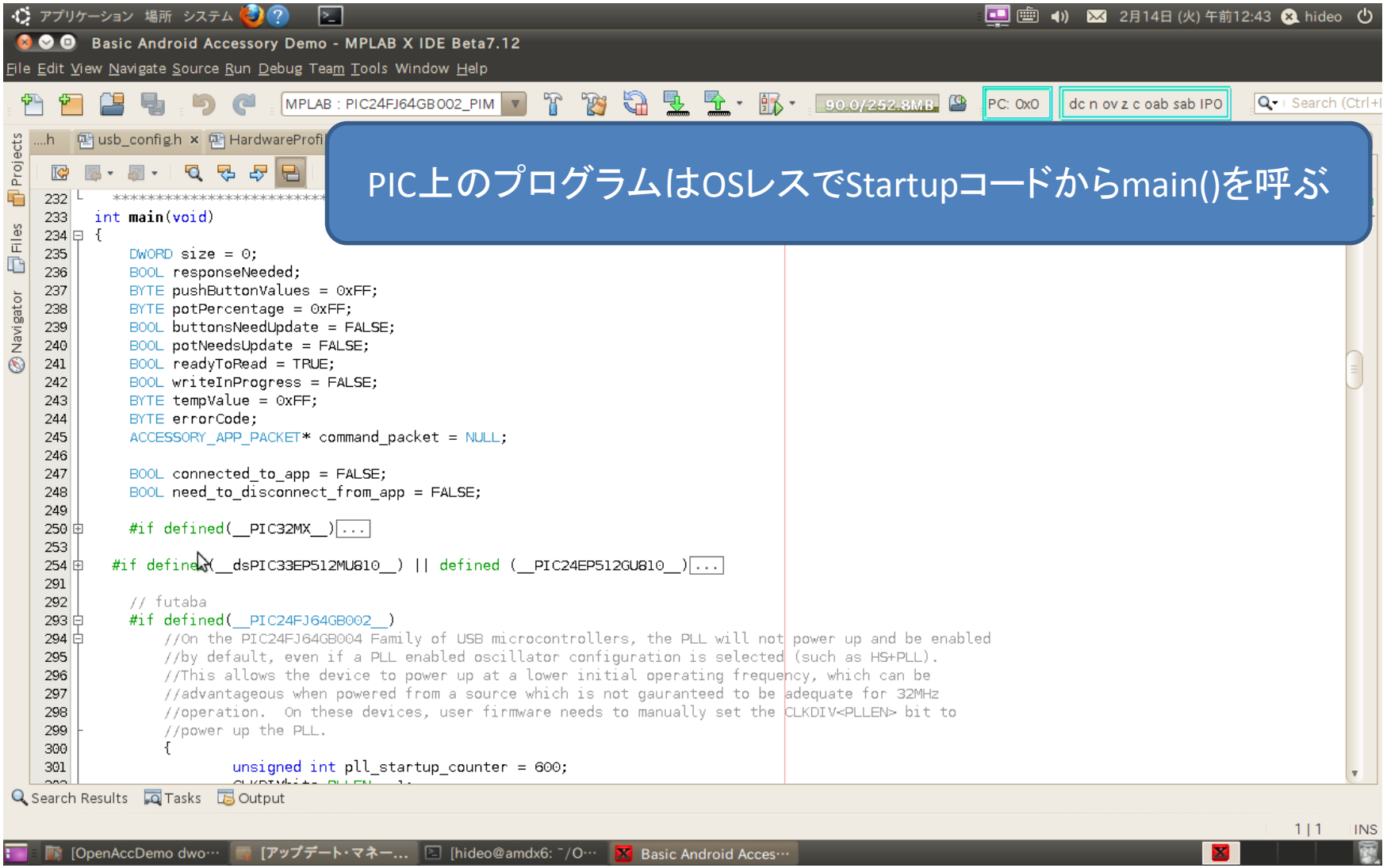


アクセサリの識別情報

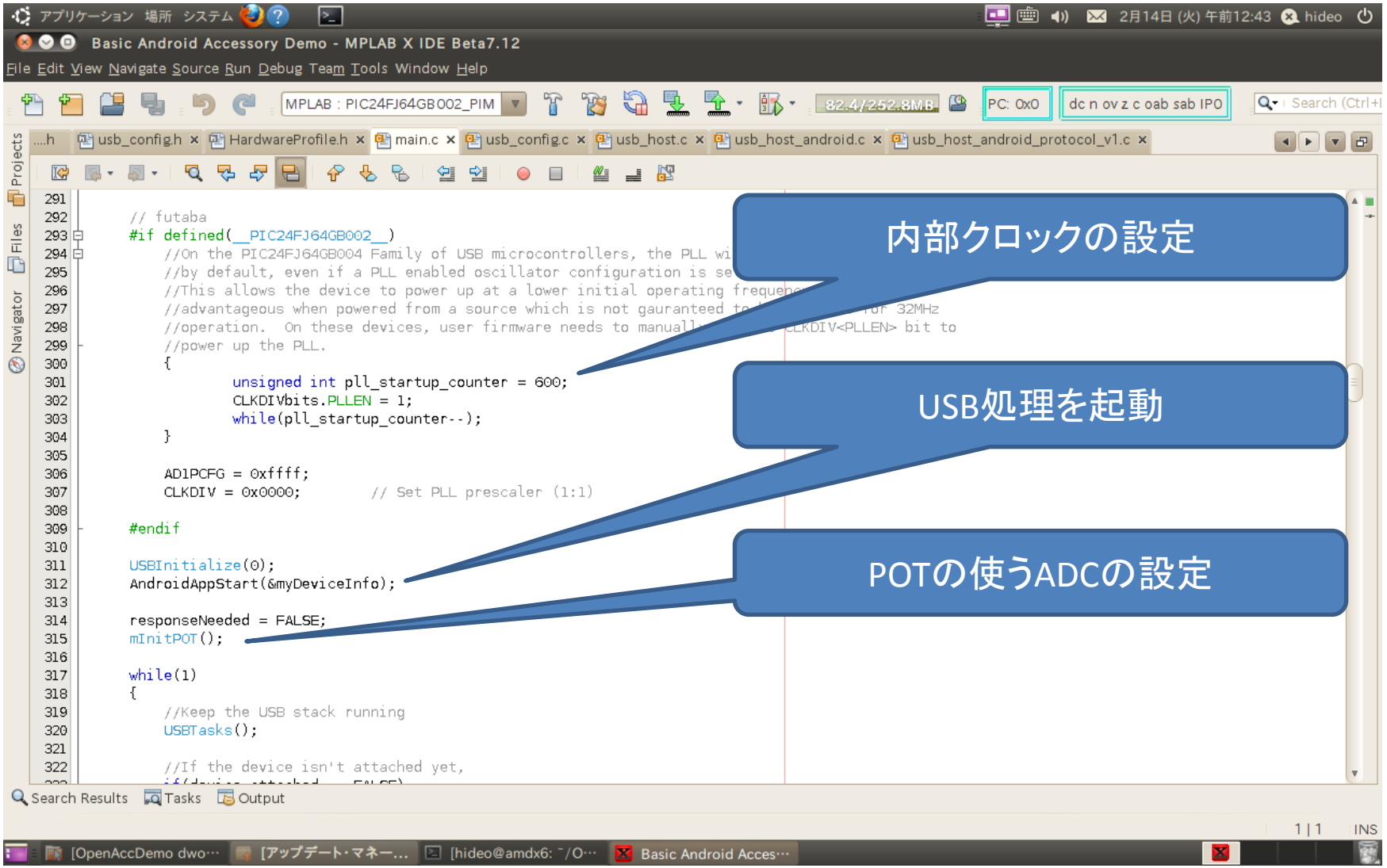



```
static char manufacturer[] = "Microchip Technology Inc.";
static char model[] = "Basic Accessory Demo";
static char description[] = DEMO_BOARD_NAME_STRING;
static char version[] = "2.0";
static char uri[] = "http://www.microchip.com/android";
static char serial[] = "N/A";
```

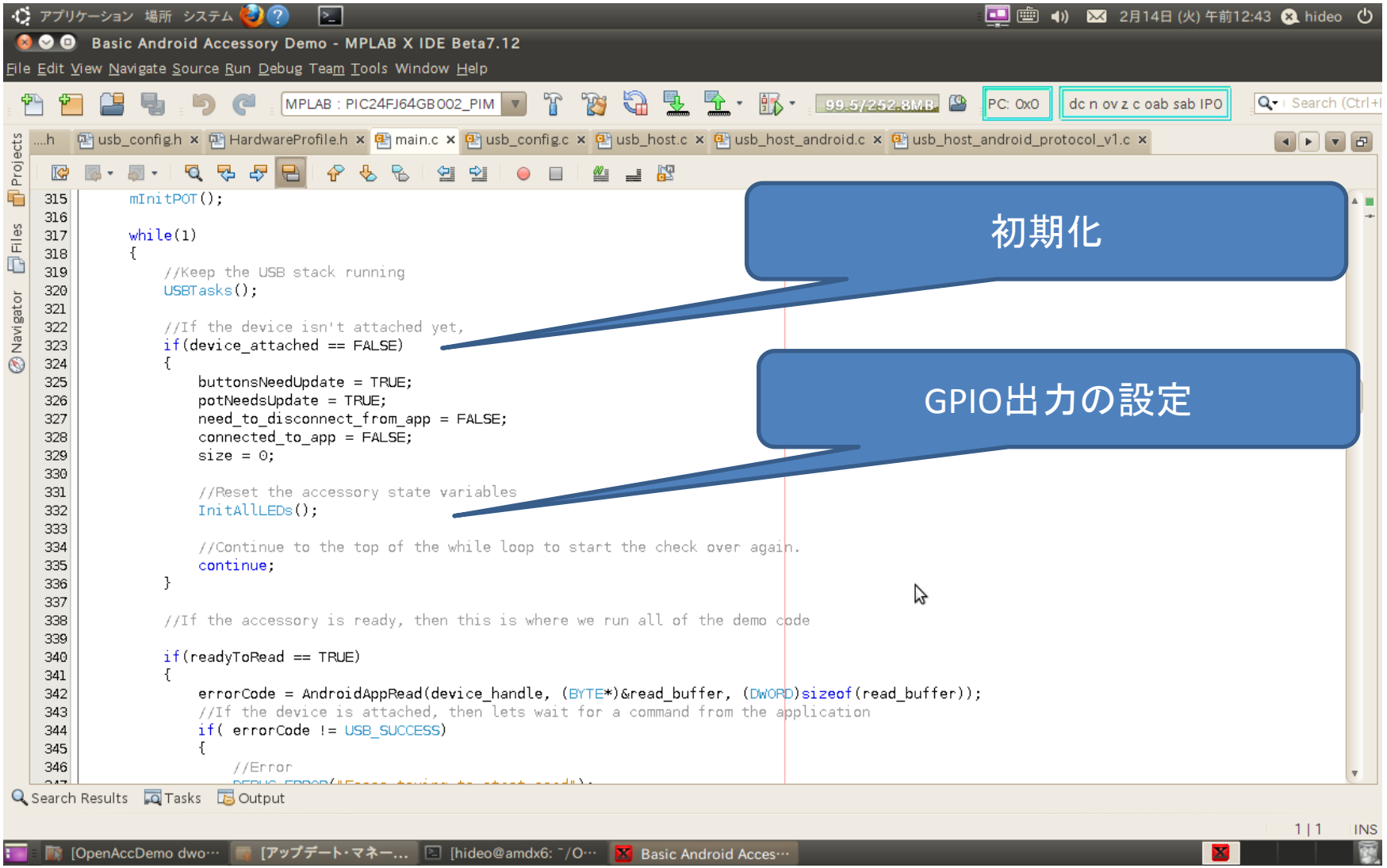
main()


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
usb_config.h
HardwareProfi
PIC上のプログラムはOSレスでStartupコードからmain()を呼ぶ
int main(void)
{
DWORD size = 0;
BOOL responseNeeded;
BYTE pushButtonValues = 0xFF;
BYTE potPercentage = 0xFF;
BOOL buttonsNeedUpdate = FALSE;
BOOL potNeedsUpdate = FALSE;
BOOL readyToRead = TRUE;
BOOL writeInProgress = FALSE;
BYTE tempValue = 0xFF;
BYTE errorCode;
ACCESSORY_APP_PACKET* command_packet = NULL;
BOOL connected_to_app = FALSE;
BOOL need_to_disconnect_from_app = FALSE;
#if defined(__PIC32MX__)...
#if defined(__dsPIC33EP512MU810__) || defined (__PIC24EP512GU810__)...
// futaba
#if defined(__PIC24FJ64GB002__)
//On the PIC24FJ64GB004 Family of USB microcontrollers, the PLL will not power up and be enabled
//by default, even if a PLL enabled oscillator configuration is selected (such as HS+PLL).
//This allows the device to power up at a lower initial operating frequency, which can be
//advantageous when powered from a source which is not gauranteed to be adequate for 32MHz
//operation. On these devices, user firmware needs to manually set the CLKDIV<PLLEN> bit to
//power up the PLL.
{
unsigned int pll_startup_counter = 600;
Search Results
Tasks
Output
Basic Android Acces…

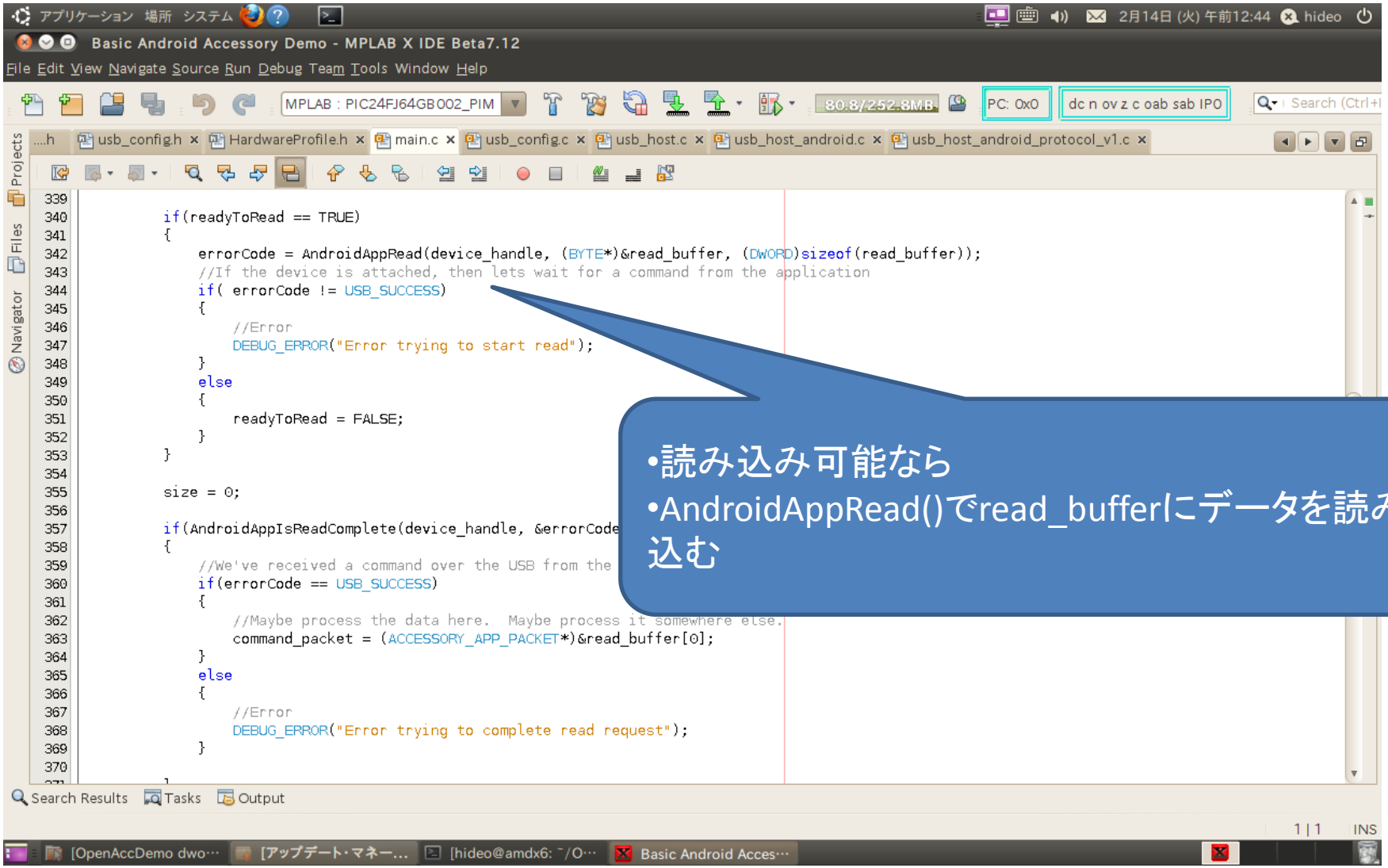
PLLの設定/ADCの初期化



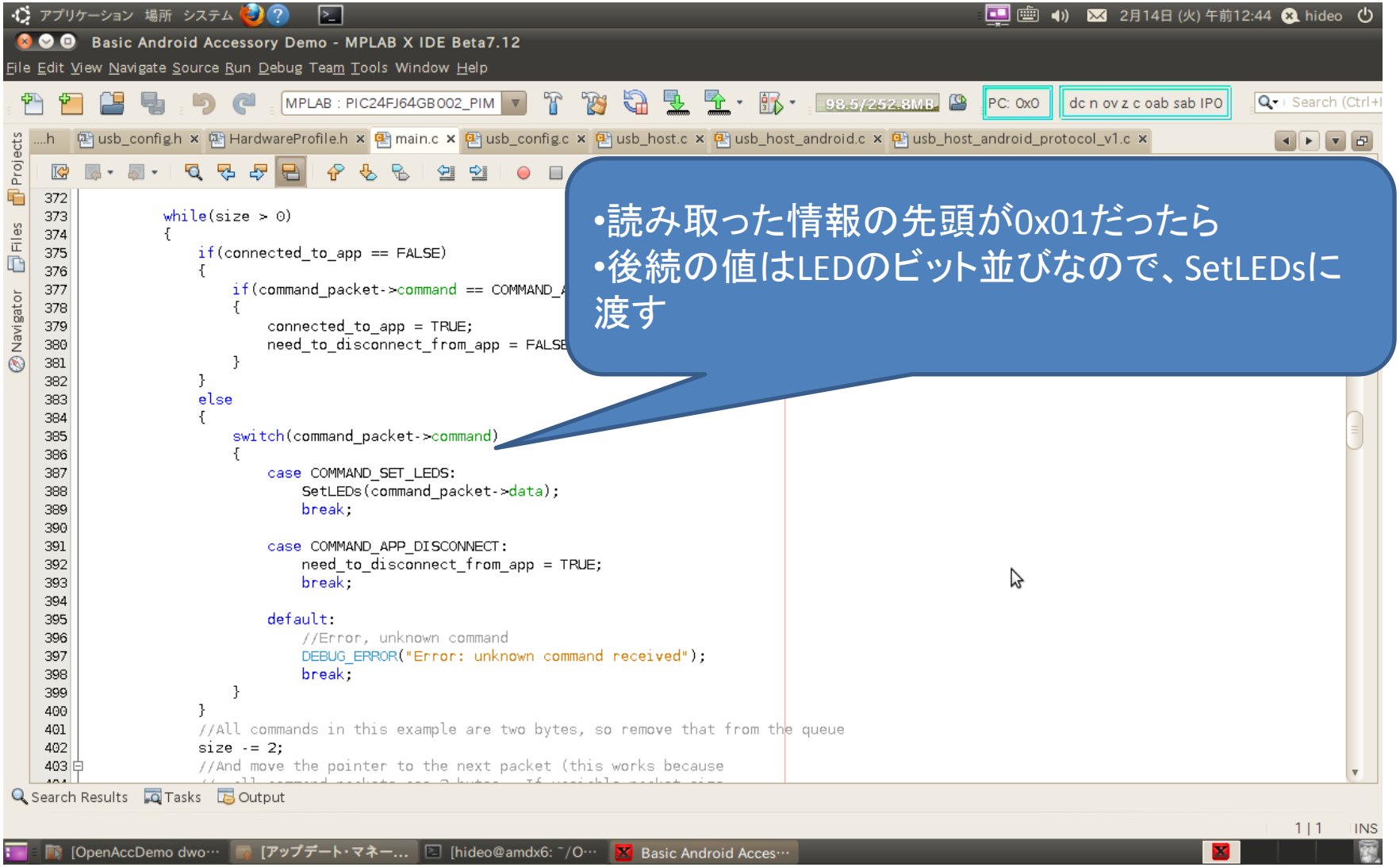
メインループの入り口


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
usb_config.h
HardwareProfile.h
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
315 mInitPOT();
316
317 while(1)
318 {
319 //Keep the USB stack running
320 USBTasks();
321
322 //If the device isn't attached yet,
323 if(device_attached == FALSE)
324 {
325 buttonsNeedUpdate = TRUE;
326 potNeedsUpdate = TRUE;
327 need_to_disconnect_from_app = FALSE;
328 connected_to_app = FALSE;
329 size = 0;
330
331 //Reset the accessory state variables
332 InitAllLEDs();
333
334 //Continue to the top of the while loop to start the check over again.
335 continue;
336 }
337
338 //If the accessory is ready, then this is where we run all of the demo code
339
340 if(readyToRead == TRUE)
341 {
342 errorCode = AndroidAppRead(device_handle, (BYTE*)&read_buffer, (DWORD)sizeof(read_buffer));
343 //If the device is attached, then lets wait for a command from the application
344 if(errorCode != USB_SUCCESS)
345 {
346 //Error
初期化
GPIO出力の設定
Search Results
Tasks
Output

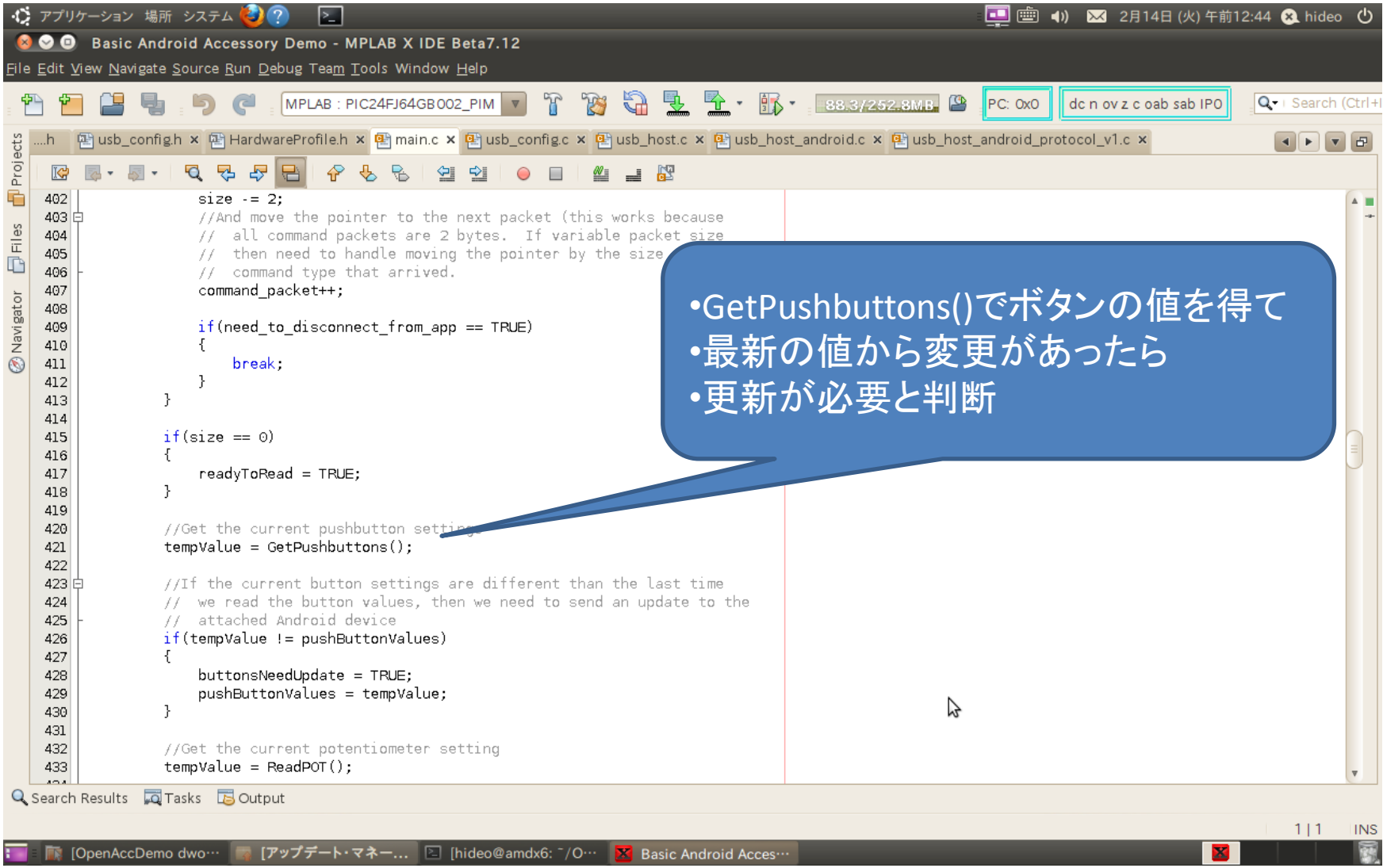
データの受信


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
usb_config.h
HardwareProfile.h
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
if(readyToRead == TRUE)
{
errorCode = AndroidAppRead(device_handle, (BYTE*)&read_buffer, (DWORD)sizeof(read_buffer));
//If the device is attached, then lets wait for a command from the application
if(errorCode != USB_SUCCESS)
{
//Error
DEBUG_ERROR("Error trying to start read");
}
else
{
readyToRead = FALSE;
}
}
size = 0;
if(AndroidAppIsReadComplete(device_handle, &errorCode
{
//We've received a command over the USB from the
if(errorCode == USB_SUCCESS)
{
//Maybe process the data here. Maybe process it somewhere else.
command_packet = (ACCESSORY_APP_PACKET*)&read_buffer[0];
}
else
{
//Error
DEBUG_ERROR("Error trying to complete read request");
}
•読み込み可能なら
•AndroidAppRead()でread_bufferにデータを読み込む
Search Results
Tasks
Output

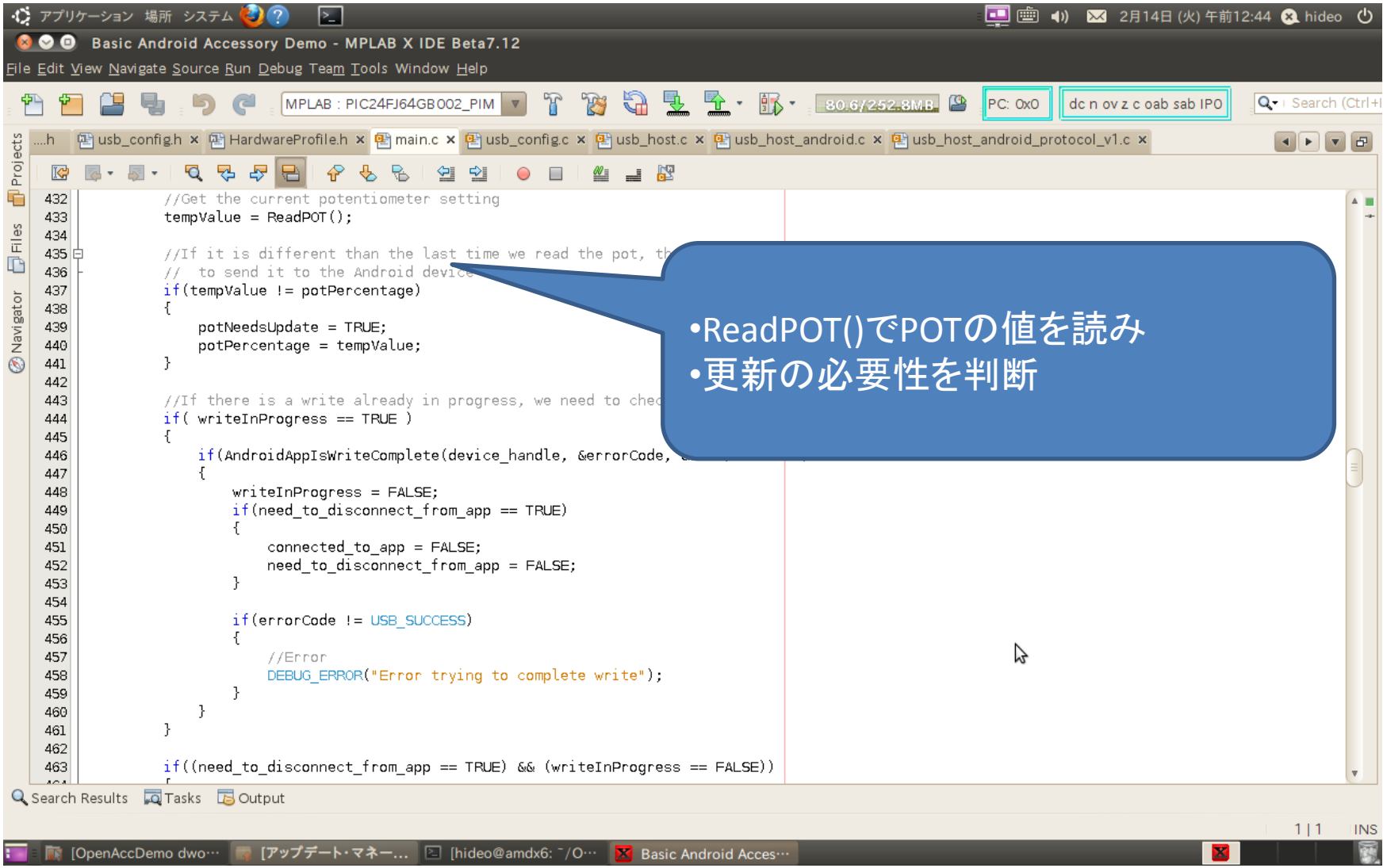
LEDの操作


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
usb_config.h
HardwareProfile.h
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
while(size > 0)
{
if(connected_to_app == FALSE)
{
if(command_packet->command == COMMAND_
{
connected_to_app = TRUE;
need_to_disconnect_from_app = FALSE
}
}
else
{
switch(command_packet->command)
{
case COMMAND_SET_LEDS:
SetLEDs(command_packet->data);
break;
case COMMAND_APP_DISCONNECT:
need_to_disconnect_from_app = TRUE;
break;
default:
//Error, unknown command
DEBUG_ERROR("Error: unknown command received");
break;
}
}
//All commands in this example are two bytes, so remove that from the queue
size -= 2;
//And move the pointer to the next packet (this works because
•読み取った情報の先頭が0x01だったら
•後続の値はLEDのビット並びなので、SetLEDsに渡す
Search Results
Tasks
Output

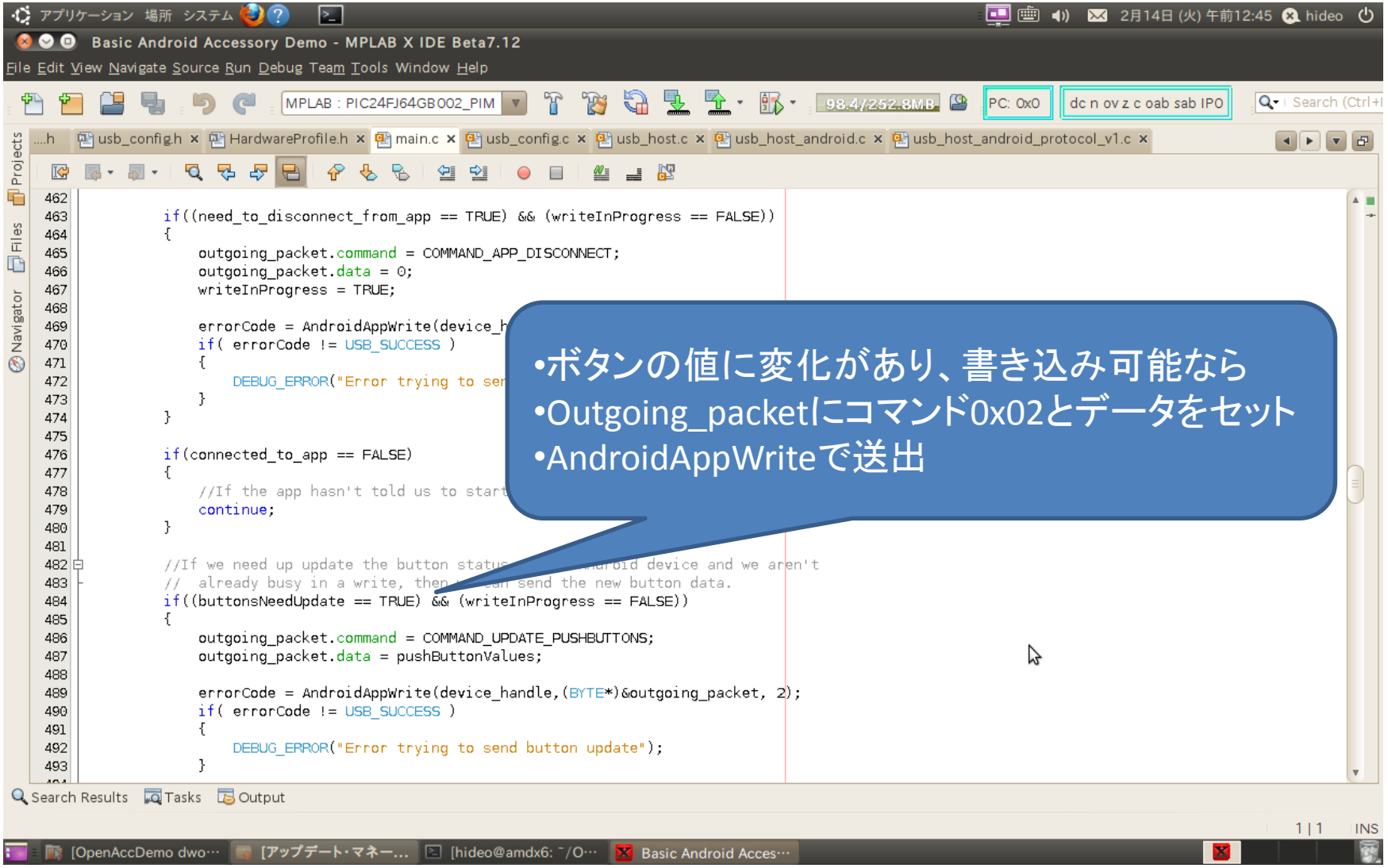
スイッチの値の読みだし



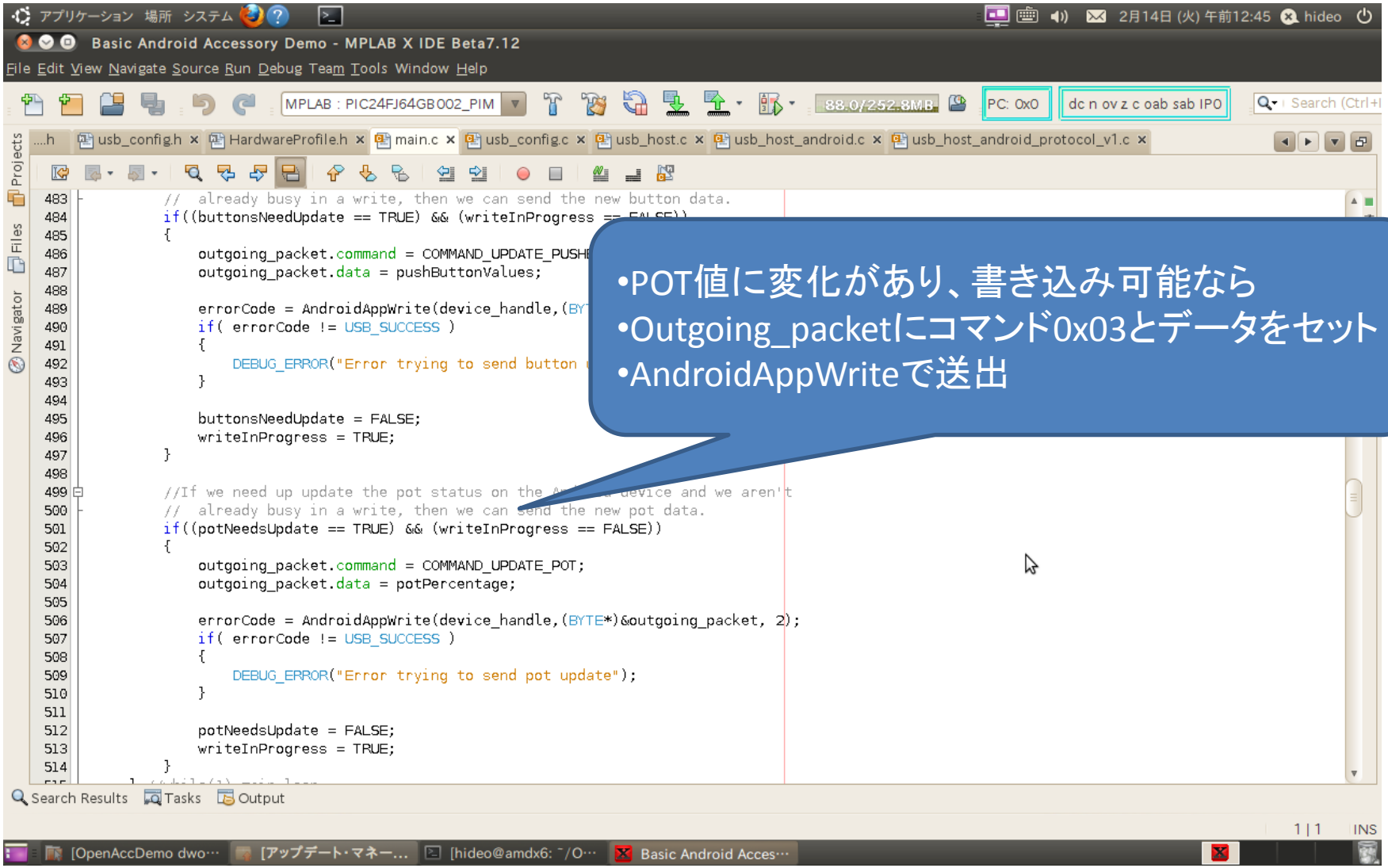
POTの値の読みだし


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
usb_config.h
HardwareProfile.h
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
//Get the current potentiometer setting
tempValue = ReadPOT();
//If it is different than the last time we read the pot, th
// to send it to the Android device
if(tempValue != potPercentage)
{
potNeedsUpdate = TRUE;
potPercentage = tempValue;
}
//If there is a write already in progress, we need to chec
if(writeInProgress == TRUE)
{
if(AndroidAppIsWriteComplete(device_handle, &errorCode,
{
writeInProgress = FALSE;
if(need_to_disconnect_from_app == TRUE)
{
connected_to_app = FALSE;
need_to_disconnect_from_app = FALSE;
}
if(errorCode != USB_SUCCESS)
{
//Error
DEBUG_ERROR("Error trying to complete write");
}
}
}
if((need_to_disconnect_from_app == TRUE) && (writeInProgress == FALSE))
•ReadPOT()でPOTの値を読み
•更新の必要性を判断
Search Results
Tasks
Output

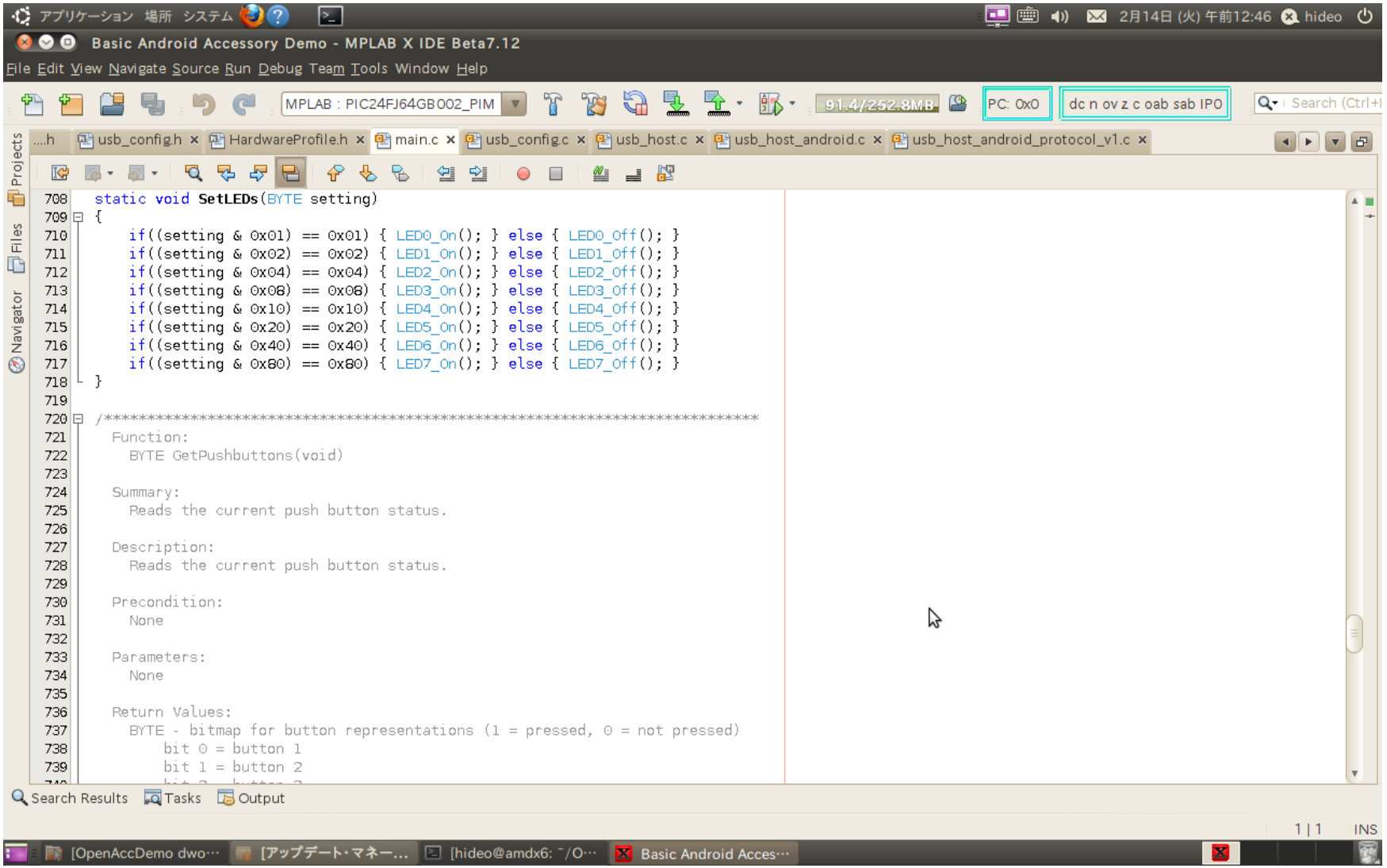
ボタンデータの送出


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
usb_config.h
HardwareProfile.h
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
if((need_to_disconnect_from_app == TRUE) && (writeInProgress == FALSE))
{
outgoing_packet.command = COMMAND_APP_DISCONNECT;
outgoing_packet.data = 0;
writeInProgress = TRUE;
errorCode = AndroidAppWrite(device_h
if(errorCode != USB_SUCCESS)
DEBUG_ERROR("Error trying to ser
if(connected_to_app == FALSE)
//If the app hasn't told us to start
continue;
//If we need up update the button statu
// already busy in a write, then
if((buttonsNeedUpdate == TRUE) && (writeInProgress == FALSE))
outgoing_packet.command = COMMAND_UPDATE_PUSHBUTTONS;
outgoing_packet.data = pushButtonValues;
errorCode = AndroidAppWrite(device_handle,(BYTE*)&outgoing_packet, 2);
if(errorCode != USB_SUCCESS)
DEBUG_ERROR("Error trying to send button update");
•ボタンの値に変化があり、書き込み可能なら
•Outgoing_packetにコマンド0x02とデータをセット
•AndroidAppWriteで送出
Search Results
Tasks
Output

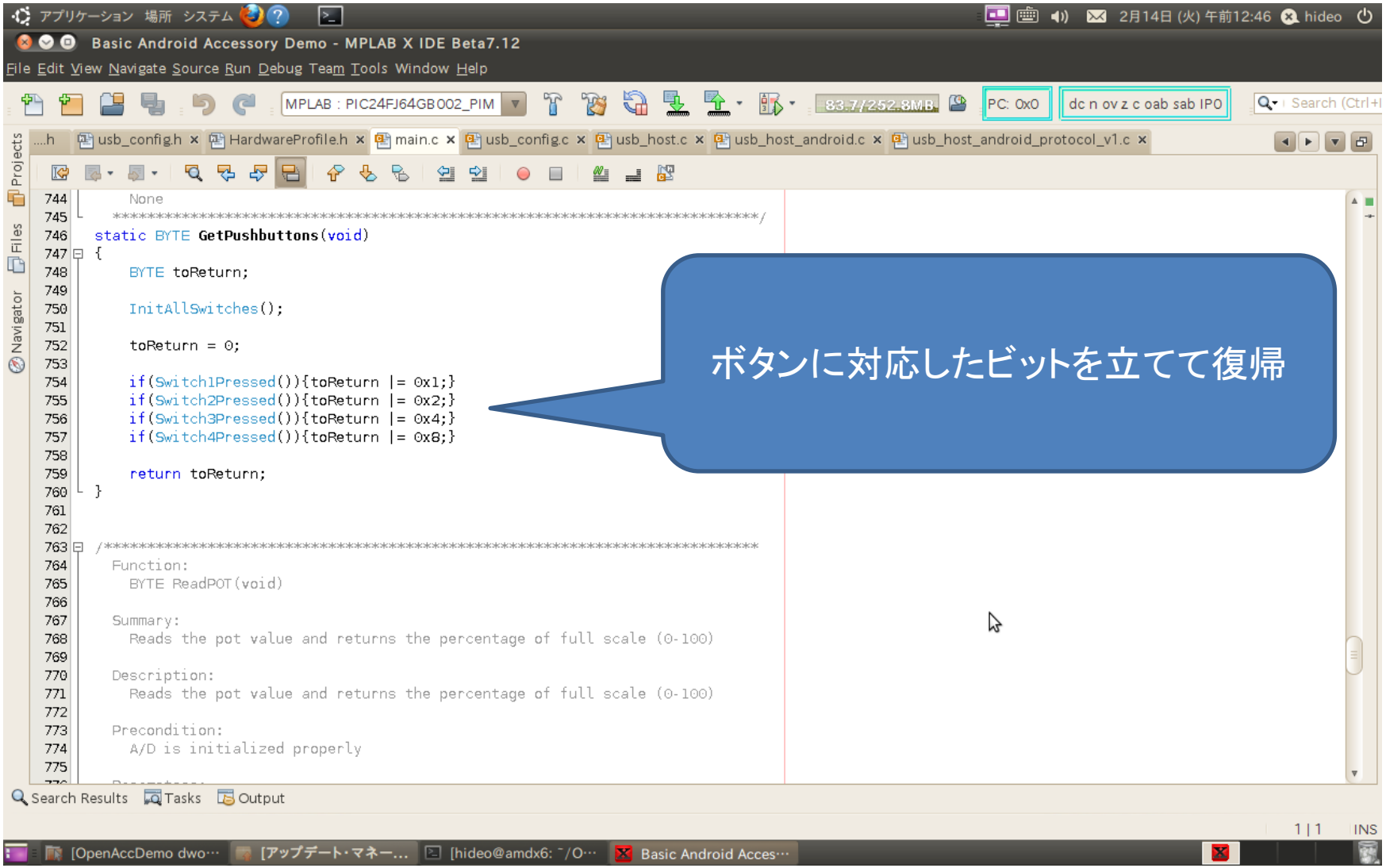
POTデータの送出


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
•POT値に変化があり、書き込み可能なら
•Outgoing_packetにコマンド0x03とデータをセット
•AndroidAppWriteで送出

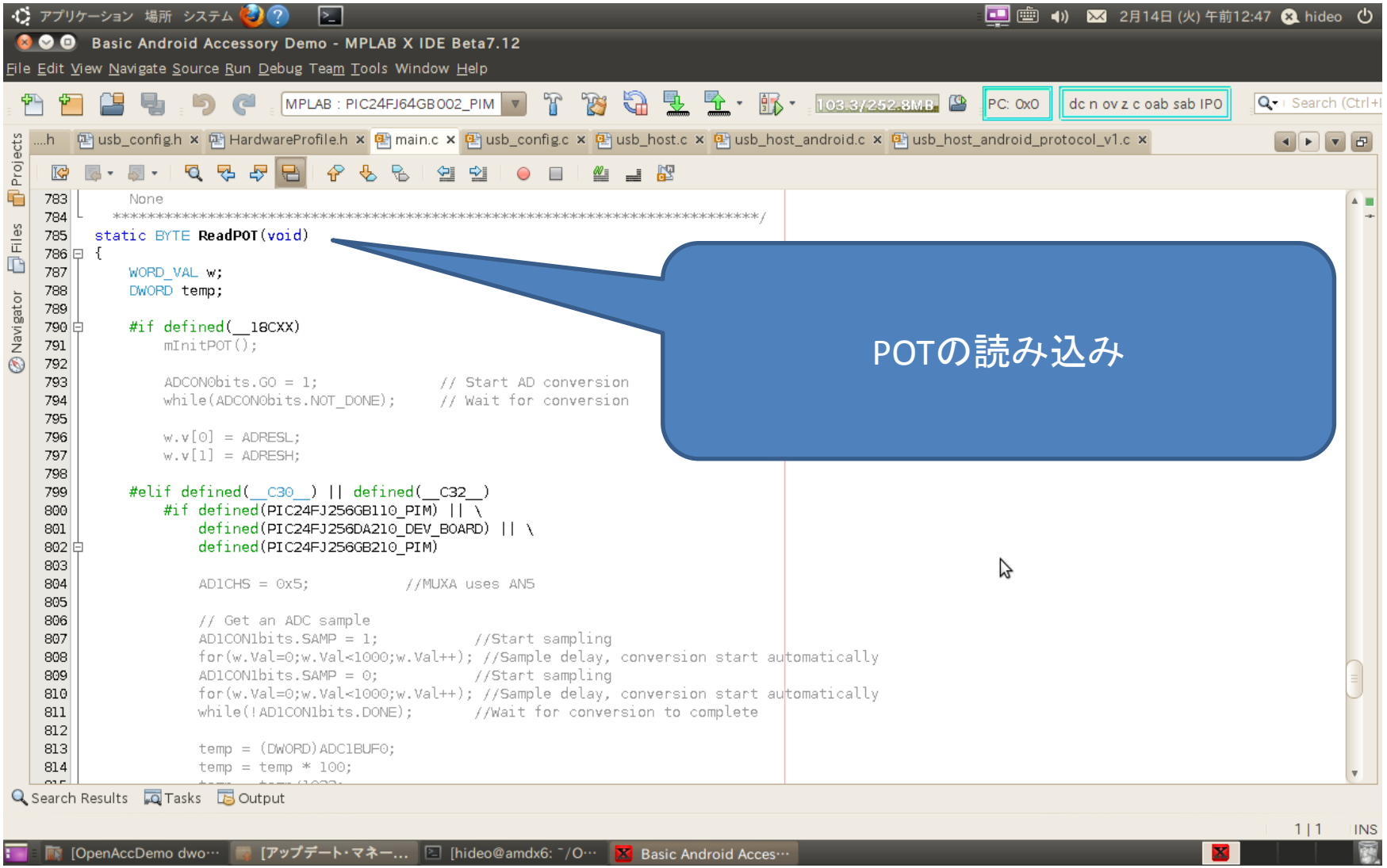
SetLEDs


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
usb_config.h
HardwareProfile.h
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
Projects
Files
Navigator
static void SetLEDs(BYTE setting)
{
if((setting & 0x01) == 0x01) { LED0_On(); } else { LED0_Off(); }
if((setting & 0x02) == 0x02) { LED1_On(); } else { LED1_Off(); }
if((setting & 0x04) == 0x04) { LED2_On(); } else { LED2_Off(); }
if((setting & 0x08) == 0x08) { LED3_On(); } else { LED3_Off(); }
if((setting & 0x10) == 0x10) { LED4_On(); } else { LED4_Off(); }
if((setting & 0x20) == 0x20) { LED5_On(); } else { LED5_Off(); }
if((setting & 0x40) == 0x40) { LED6_On(); } else { LED6_Off(); }
if((setting & 0x80) == 0x80) { LED7_On(); } else { LED7_Off(); }
}
/***
Function:
BYTE GetPushbuttons(void)
Summary:
Reads the current push button status.
Description:
Reads the current push button status.
Precondition:
None
Parameters:
None
Return Values:
BYTE - bitmap for button representations (1 = pressed, 0 = not pressed)
bit 0 = button 1
bit 1 = button 2
Search Results
Tasks
Output
Basic Android Acces…

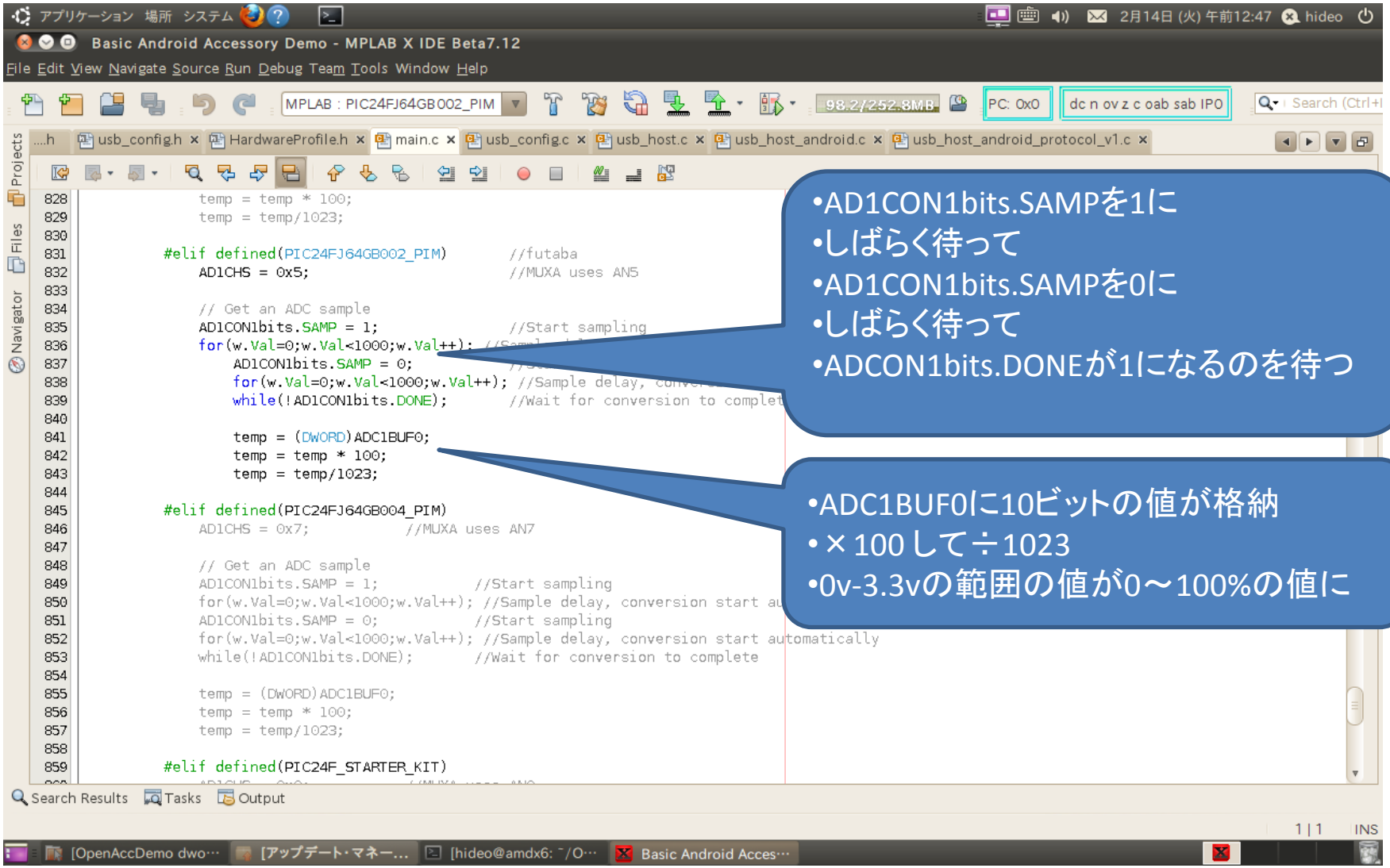
GetPushbuttons



POTの読み取り


Basic Android Accessory Demo - MPLAB X IDE Beta7.12
File Edit View Navigate Source Run Debug Team Tools Window Help
MPLAB : PIC24FJ64GB002_PIM
usb_config.h
HardwareProfile.h
main.c
usb_config.c
usb_host.c
usb_host_android.c
usb_host_android_protocol_v1.c
783 None
784 ********/
785 static BYTE ReadPOT(void)
786 {
787 WORD_VAL w;
788 DWORD temp;
789
790 #if defined(__18CXX)
791 mInitPOT();
792
793 ADCON0bits.GO = 1; // Start AD conversion
794 while(ADCON0bits.NOT_DONE); // Wait for conversion
795
796 w.v[0] = ADRESL;
797 w.v[1] = ADRESH;
798
799 #elif defined(__C30__) || defined(__C32__)
800 #if defined(PIC24FJ256GB110_PIM) || \
801 defined(PIC24FJ256DA210_DEV_BOARD) || \
802 defined(PIC24FJ256GB210_PIM)
803
804 AD1CHS = 0x5; //MUXA uses AN5
805
806 // Get an ADC sample
807 AD1CON1bits.SAMP = 1; //Start sampling
808 for(w.Val=0;w.Val<1000;w.Val++); //Sample delay, conversion start automatically
809 AD1CON1bits.SAMP = 0; //Start sampling
810 for(w.Val=0;w.Val<1000;w.Val++); //Sample delay, conversion start automatically
811 while(!AD1CON1bits.DONE); //Wait for conversion to complete
812
813 temp = (DWORD)ADC1BUF0;
814 temp = temp * 100;
POTの読み込み
Search Results
Tasks
Output

ADCの起動と読み込み



PART5
ANDROID側の仕組み

Android USB Accessory

- Android 2.3.4(API Level10)にバックポート
  - com.android.future.usb
  - add-on Google APIs Android API 10が必要
  - http://code.google.com/intl/ja/android/add-ons/google-apis/installing.html
- Android 3.1 (API level 12)でサポート
  - Android.hardware.usb

Manifiest

- Android 2.3.4(API Level10) + add-on
  - <uses-library>エレメント
  - com.android.feature.usb.accessory
- Android 3.1 (API level 12)
  - <uses-feature>エレメント
  - android.hardware.usb.accessory

API

- Android 2.3.4(API Level10) + add-on
  - UsbManager manager = UsbManager.getInstance(**this**);
  - UsbAccessory accessory = UsbManager.getAccessory(intent);
- Android 3.1 (API level 12)
  - UsbManager manager = (UsbManager) getSystemService(Context.USB_SERVICE);
  - UsbAccessory accessory = (UsbAccessory) intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);

接続の通知

- android.hardware.usb.action.USB_ACCESSORY_ATTACHED
  - メインのActivityに次の２つをペアで記述
    - <intent-filter>　　インテントフィルタ
    - <meta-data>　　アクセサリ識別のXMLファイル
- XMLファイルの記述
  - <usb-accessory>エレメントに以下を記述
    - manufacturer
    - model
    - version

UsbAccessoryの取得

- Android 2.3.4(API Level10) + add-on

  UsbAccessory accessory = UsbManager.getAccessor
  y(intent);

- Android 3.1 (API level 12)

  UsbAccessory accessory = (UsbAccessory)intent.get
  ParcelableExtra(UsbManager.EXTRA_ACCESSORY);

Accessoriesの列挙

- Android 2.3.4(API Level10) + add-on

  UsbManager manager = UsbManager.getInstance(**this**);

  UsbAccessory[] accessoryList = manager.getAcccessoryList();

- Android 3.1 (API level 12)

  UsbManager manager = (UsbManager) getSystemService(Co
  ntext.USB_SERVICE);

  UsbAccessory[] accessoryList = manager.getAcccessoryList();

通信の許可

- broadcast receiver の作成
- Activity の onCreate() にbroadcast receiver を登録
- パーミッションダイアログの表示

```
UsbAccessory accessory;
...
mUsbManager.requestPermission(accessory, mPermissionIntent);
```

Accessoryとの通信

- UsbManagerでfile descripterを取得
  - InputStream/OutputStreamがEndpointに接続されている
- 通信はブロックされるの
  - スレッドを起こして、run()メソッドで通信

APIのまとめ

- 2.x系と3-4系でAPIが異なる
  - 3-4系はframeworkに実装
  - 2.xではadd-onのlibraryで実装
- Manifestに記述する
  - 3-4系はuses-feature, 2.x uses-library
- UsbManagerとUsbAccessory を取得する
- Readは同期IOなので、別スレッドで呼ぶ

Android Appソースのウォークスルー

- AndroidManifest.xml
- res/xml/accessory_filter.xml
- src/com/microchip/android/BasicAccessoryDemo_API12

```
$ wc -l *
  512 BasicAccessoryDemo.java          アプリ本体
  138 LEDControl.java                  LED描画
  854 USBAccessoryManager.java
   93 USBAccessoryManagerMessage.java
 1597 合計
```

LifeCycle handler

- onCreate

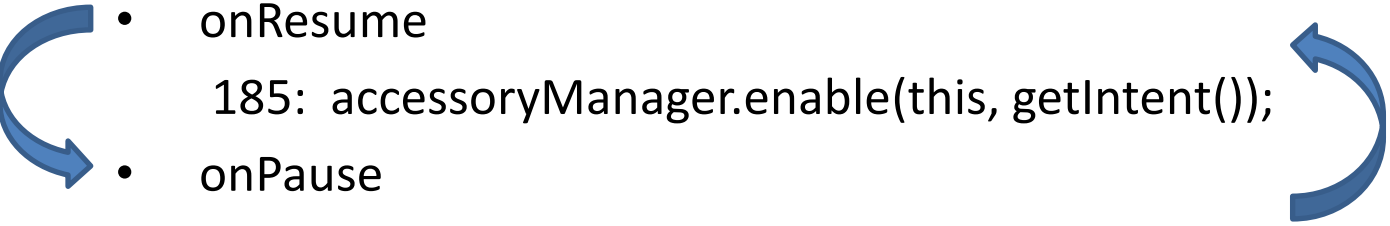
  92: accessoryManager =

  new USBAccessoryManager(handler, USBAccessoryWhat);
- onStart

  177: this.setTitle("Basic Accessory Demo: Device not connected.");
- onResume

  185: accessoryManager.enable(this, getIntent());
- onPause

  242: accessoryManager.disable(this);



USBAccessoryManager.java

USBAccessoryManager.enable

- UsbManagerからUsbAccessoryを取得

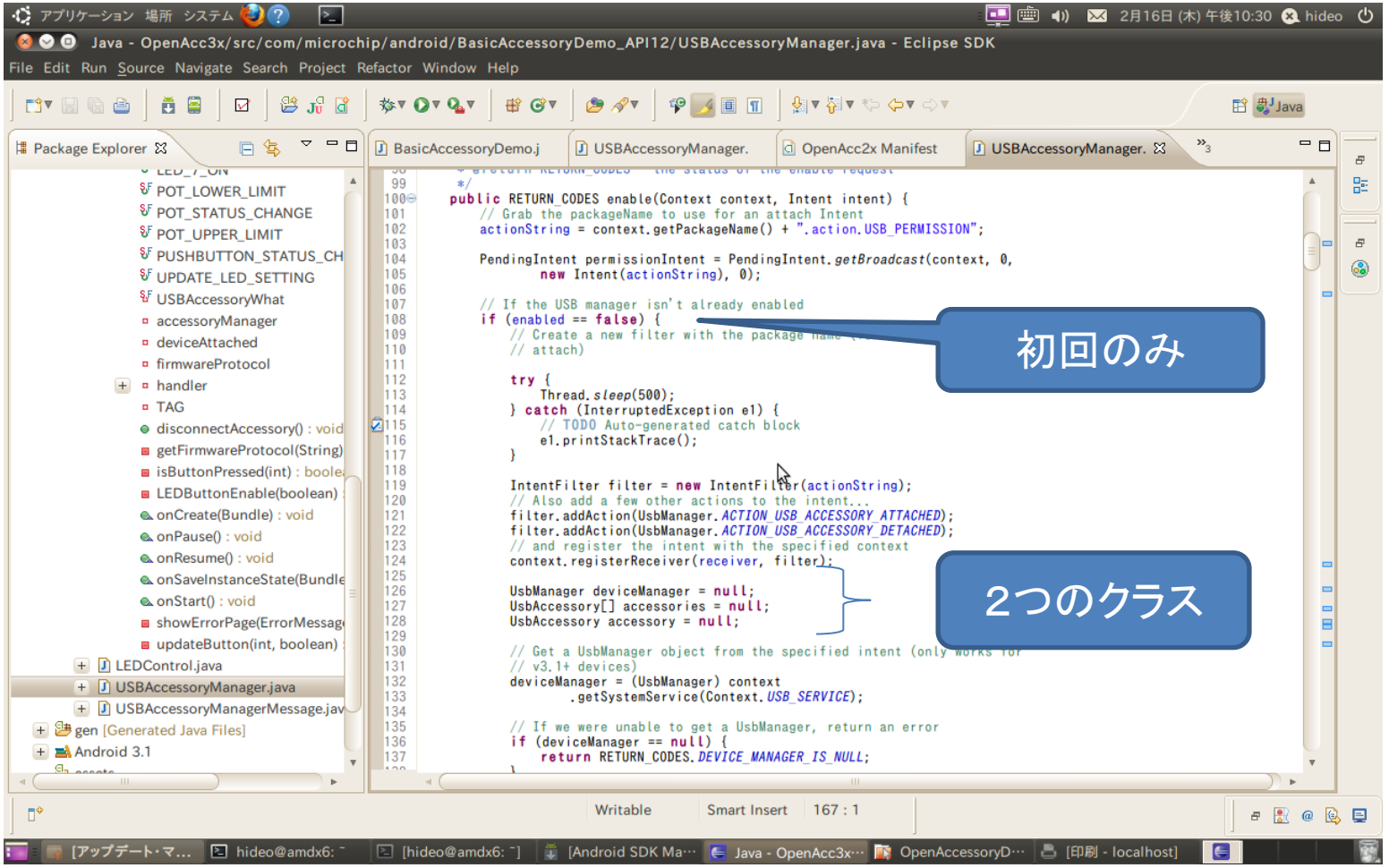
  141: accessories = deviceManager.getAccessoryList();
- ファイルディスクリプタを取得

  159: parcelFileDescriptor = deviceManager.openAccessory(accessory);
- ReadTreadを生成

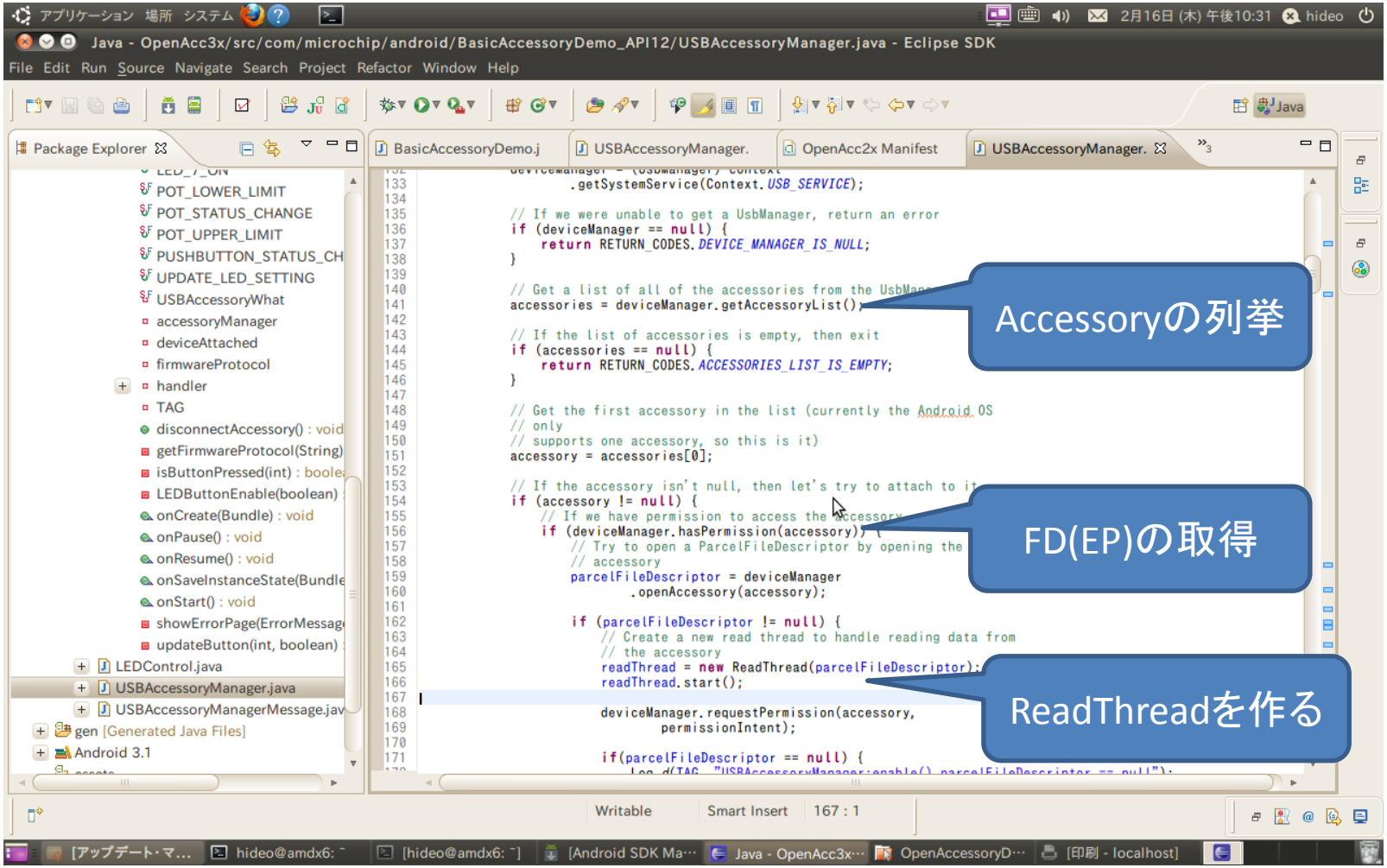
  165: readThread = new ReadThread(parcelFileDescriptor);
- outputStreamを取得

  178: outputStream = new
  FileOutputStream(parcelFileDescriptor.getFileDescriptor());

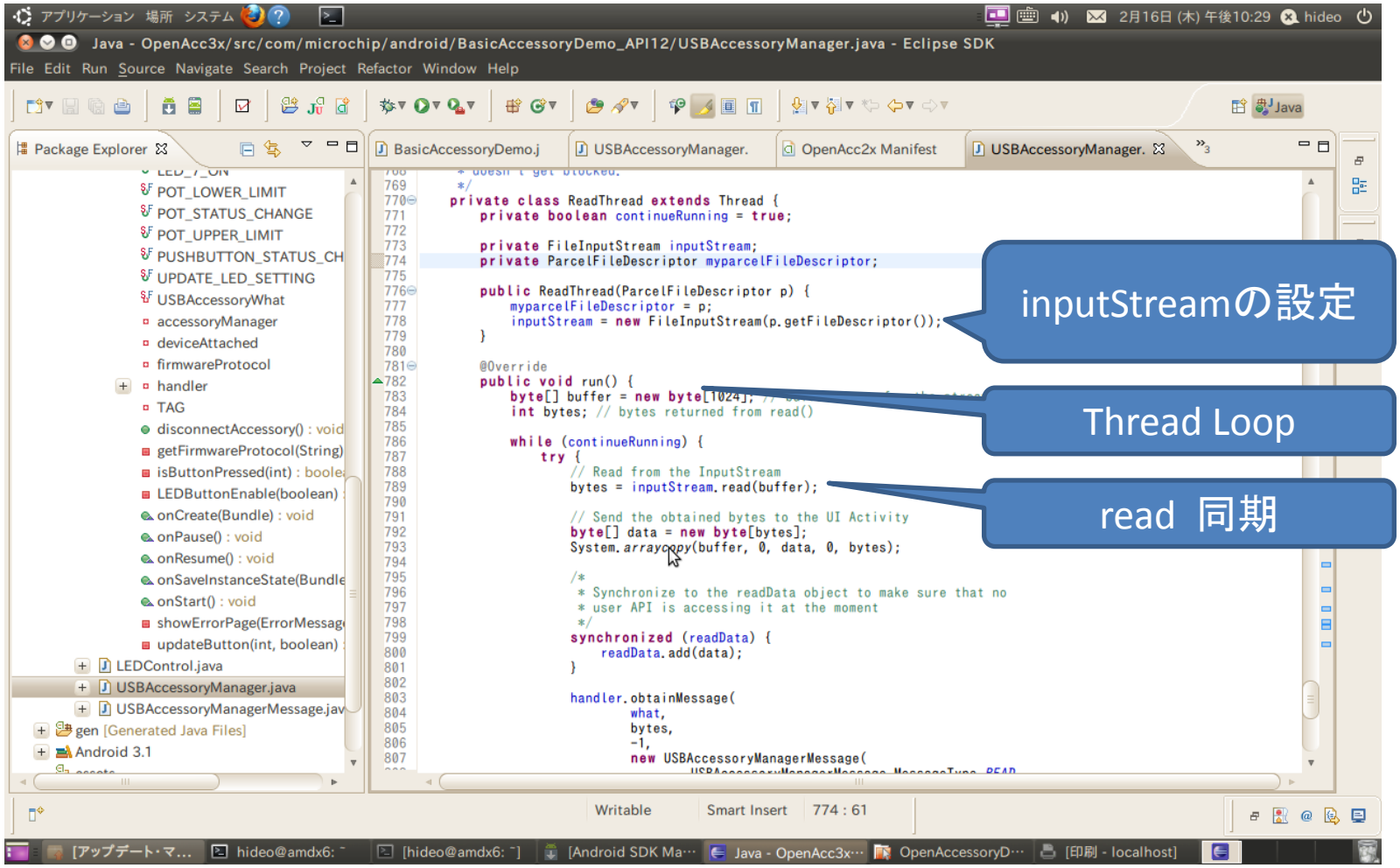
enable - 1


Java - OpenAcc3x/src/com/microchip/android/BasicAccessoryDemo_API12/USBAccessoryManager.java - Eclipse SDK
File Edit Run Source Navigate Search Project Refactor Window Help
Package Explorer
BasicAccessoryDemo.j
USBAccessoryManager.
OpenAcc2x Manifest
USBAccessoryManager.
public RETURN_CODES enable(Context context, Intent intent) {
// Grab the packageName to use for an attach Intent
actionString = context.getPackageName() + ".action.USB_PERMISSION";
// If the USB manager isn't already enabled
if (enabled == false) {
IntentFilter filter = new IntentFilter(actionString);
// Also add a few other actions to the intent...
filter.addAction(UsbManager.ACTION_USB_ACCESSORY_ATTACHED);
filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
// and register the intent with the specified context
context.registerReceiver(receiver, filter);
UsbManager deviceManager = null;
UsbAccessory[] accessories = null;
UsbAccessory accessory = null;
// Get a UsbManager object from the specified intent (only works for
// v3.1+ devices)
deviceManager = (UsbManager) context
.getSystemService(Context.USB_SERVICE);
// If we were unable to get a UsbManager, return an error
if (deviceManager == null) {
return RETURN_CODES.DEVICE_MANAGER_IS_NULL;
初回のみ
2つのクラス
Writable
Smart Insert
167 : 1

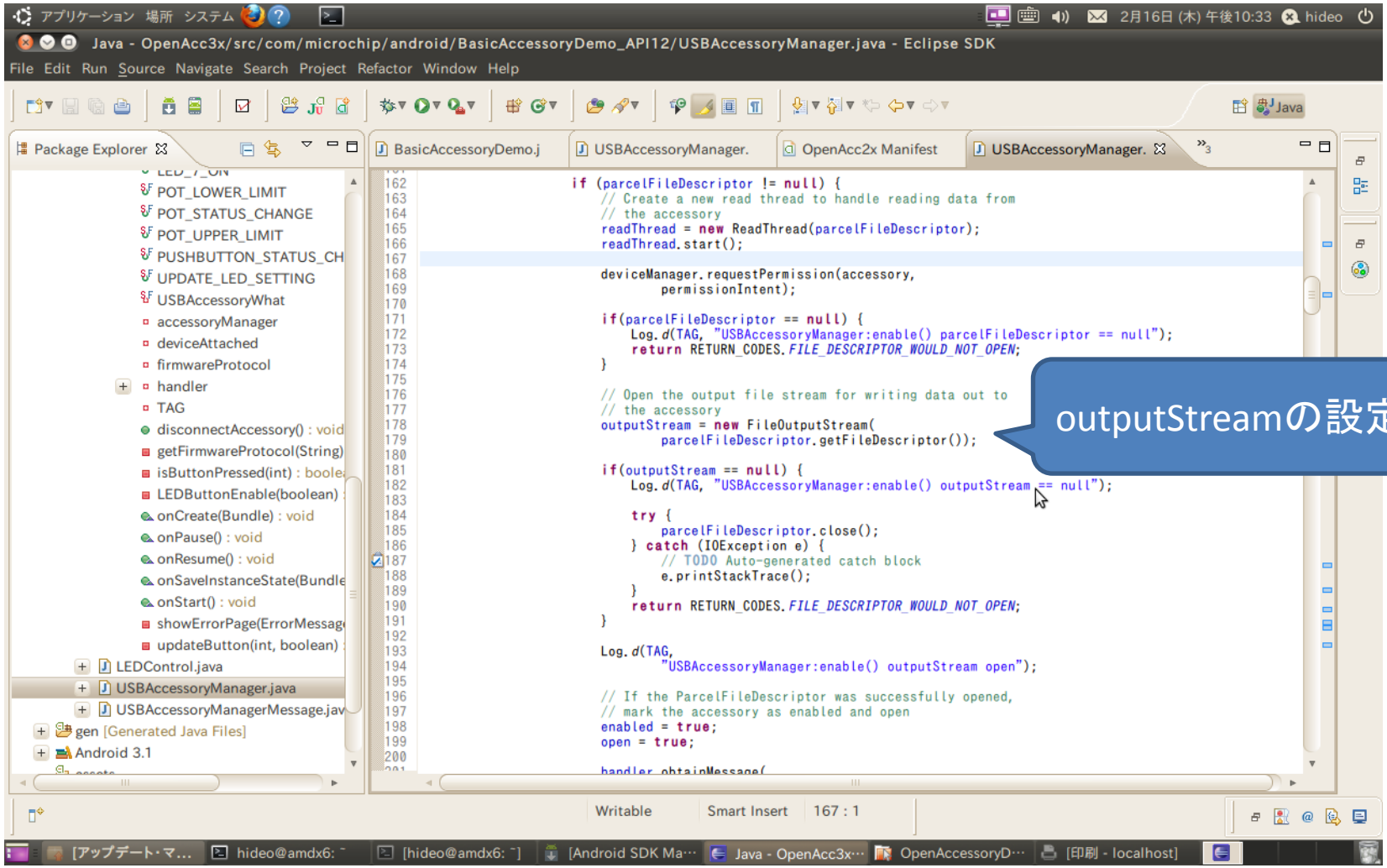
enable - 2


Java - OpenAcc3x/src/com/microchip/android/BasicAccessoryDemo_API12/USBAccessoryManager.java - Eclipse SDK
File Edit Run Source Navigate Search Project Refactor Window Help
Package Explorer
BasicAccessoryDemo.j
USBAccessoryManager.
OpenAcc2x Manifest
USBAccessoryManager.
POT_LOWER_LIMIT
POT_STATUS_CHANGE
POT_UPPER_LIMIT
PUSHBUTTON_STATUS_CH
UPDATE_LED_SETTING
USBAccessoryWhat
accessoryManager
deviceAttached
firmwareProtocol
handler
TAG
disconnectAccessory() : void
getFirmwareProtocol(String)
isButtonPressed(int) : boole
LEDButtonEnable(boolean)
onCreate(Bundle) : void
onPause() : void
onResume() : void
onSaveInstanceState(Bundle
onStart() : void
showErrorPage(ErrorMessag
updateButton(int, boolean)
LEDControl.java
USBAccessoryManager.java
USBAccessoryManagerMessage.jav
gen [Generated Java Files]
Android 3.1
.getSystemService(Context.USB_SERVICE);
// If we were unable to get a UsbManager, return an error
if (deviceManager == null) {
return RETURN_CODES.DEVICE_MANAGER_IS_NULL;
}
// Get a list of all of the accessories from the UsbMana
accessories = deviceManager.getAccessoryList();
// If the list of accessories is empty, then exit
if (accessories == null) {
return RETURN_CODES.ACCESSORIES_LIST_IS_EMPTY;
}
// Get the first accessory in the list (currently the Android OS
// only
// supports one accessory, so this is it)
accessory = accessories[0];
// If the accessory isn't null, then let's try to attach to it
if (accessory != null) {
// If we have permission to access the accessory
if (deviceManager.hasPermission(accessory)) {
// Try to open a ParcelFileDescriptor by opening the
// accessory
parcelFileDescriptor = deviceManager
.openAccessory(accessory);
if (parcelFileDescriptor != null) {
// Create a new read thread to handle reading data from
// the accessory
readThread = new ReadThread(parcelFileDescriptor);
readThread.start();
deviceManager.requestPermission(accessory,
permissionIntent);
if(parcelFileDescriptor == null) {
Accessoryの列挙
FD(EP)の取得
ReadThreadを作る
Writable
Smart Insert
167 : 1

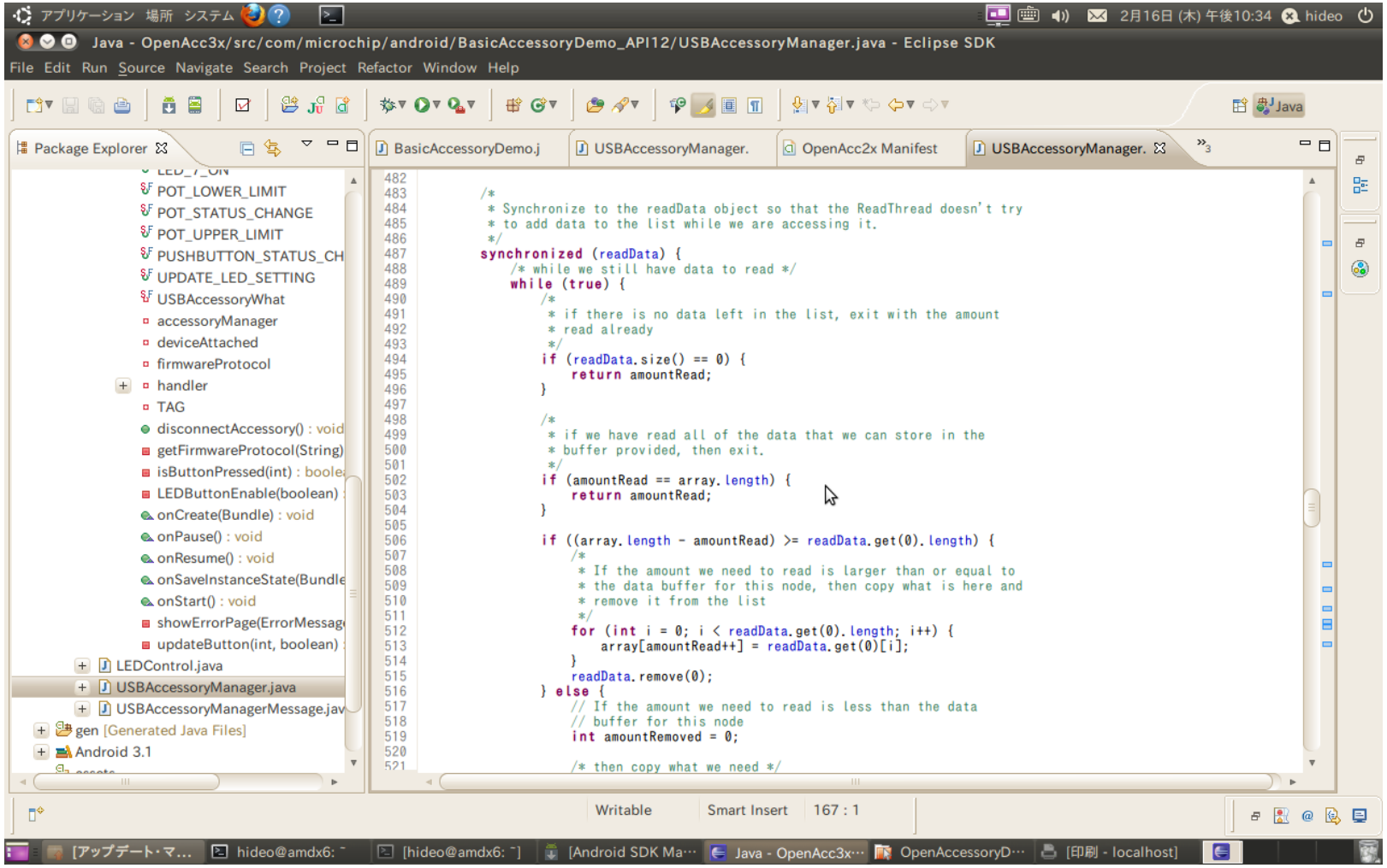
readThread


Java - OpenAcc3x/src/com/microchip/android/BasicAccessoryDemo_API12/USBAccessoryManager.java - Eclipse SDK
inputStreamの設定
Thread Loop
read 同期

enable - 3


Java - OpenAcc3x/src/com/microchip/android/BasicAccessoryDemo_API12/USBAccessoryManager.java - Eclipse SDK
File Edit Run Source Navigate Search Project Refactor Window Help
Package Explorer
BasicAccessoryDemo.j
USBAccessoryManager.
OpenAcc2x Manifest
USBAccessoryManager.
POT_LOWER_LIMIT
POT_STATUS_CHANGE
POT_UPPER_LIMIT
PUSHBUTTON_STATUS_CH
UPDATE_LED_SETTING
USBAccessoryWhat
accessoryManager
deviceAttached
firmwareProtocol
handler
TAG
disconnectAccessory() : void
getFirmwareProtocol(String)
isButtonPressed(int) : boole
LEDButtonEnable(boolean)
onCreate(Bundle) : void
onPause() : void
onResume() : void
onSaveInstanceState(Bundle
onStart() : void
showErrorPage(ErrorMessag
updateButton(int, boolean)
LEDControl.java
USBAccessoryManager.java
USBAccessoryManagerMessage.jav
gen [Generated Java Files]
Android 3.1
if (parcelFileDescriptor != null) {
// Create a new read thread to handle reading data from
// the accessory
readThread = new ReadThread(parcelFileDescriptor);
readThread.start();
deviceManager.requestPermission(accessory,
permissionIntent);
if(parcelFileDescriptor == null) {
Log.d(TAG, "USBAccessoryManager:enable() parcelFileDescriptor == null");
return RETURN_CODES.FILE_DESCRIPTOR_WOULD_NOT_OPEN;
}
// Open the output file stream for writing data out to
// the accessory
outputStream = new FileOutputStream(
parcelFileDescriptor.getFileDescriptor());
if(outputStream == null) {
Log.d(TAG, "USBAccessoryManager:enable() outputStream == null");
try {
parcelFileDescriptor.close();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
return RETURN_CODES.FILE_DESCRIPTOR_WOULD_NOT_OPEN;
}
Log.d(TAG,
"USBAccessoryManager:enable() outputStream open");
// If the ParcelFileDescriptor was successfully opened,
// mark the accessory as enabled and open
enabled = true;
open = true;
outputStreamの設定
Writable
Smart Insert
167 : 1

enable - 4


Java - OpenAcc3x/src/com/microchip/android/BasicAccessoryDemo_API12/USBAccessoryManager.java - Eclipse SDK
File Edit Run Source Navigate Search Project Refactor Window Help
Package Explorer
BasicAccessoryDemo.j
USBAccessoryManager.
OpenAcc2x Manifest
USBAccessoryManager.
POT_LOWER_LIMIT
POT_STATUS_CHANGE
POT_UPPER_LIMIT
PUSHBUTTON_STATUS_CH
UPDATE_LED_SETTING
USBAccessoryWhat
accessoryManager
deviceAttached
firmwareProtocol
handler
TAG
disconnectAccessory() : void
getFirmwareProtocol(String)
isButtonPressed(int) : boole
LEDButtonEnable(boolean)
onCreate(Bundle) : void
onPause() : void
onResume() : void
onSaveInstanceState(Bundle
onStart() : void
showErrorPage(ErrorMessag
updateButton(int, boolean)
LEDControl.java
USBAccessoryManager.java
USBAccessoryManagerMessage.jav
gen [Generated Java Files]
Android 3.1
/*
* Synchronize to the readData object so that the ReadThread doesn't try
* to add data to the list while we are accessing it.
*/
synchronized (readData) {
/* while we still have data to read */
while (true) {
/*
* if there is no data left in the list, exit with the amount
* read already
*/
if (readData.size() == 0) {
return amountRead;
}
/*
* if we have read all of the data that we can store in the
* buffer provided, then exit.
*/
if (amountRead == array.length) {
return amountRead;
}
if ((array.length - amountRead) >= readData.get(0).length) {
/*
* If the amount we need to read is larger than or equal to
* the data buffer for this node, then copy what is here and
* remove it from the list
*/
for (int i = 0; i < readData.get(0).length; i++) {
array[amountRead++] = readData.get(0)[i];
}
readData.remove(0);
} else {
// If the amount we need to read is less than the data
// buffer for this node
int amountRemoved = 0;
/* then copy what we need */
Writable
Smart Insert
167 : 1

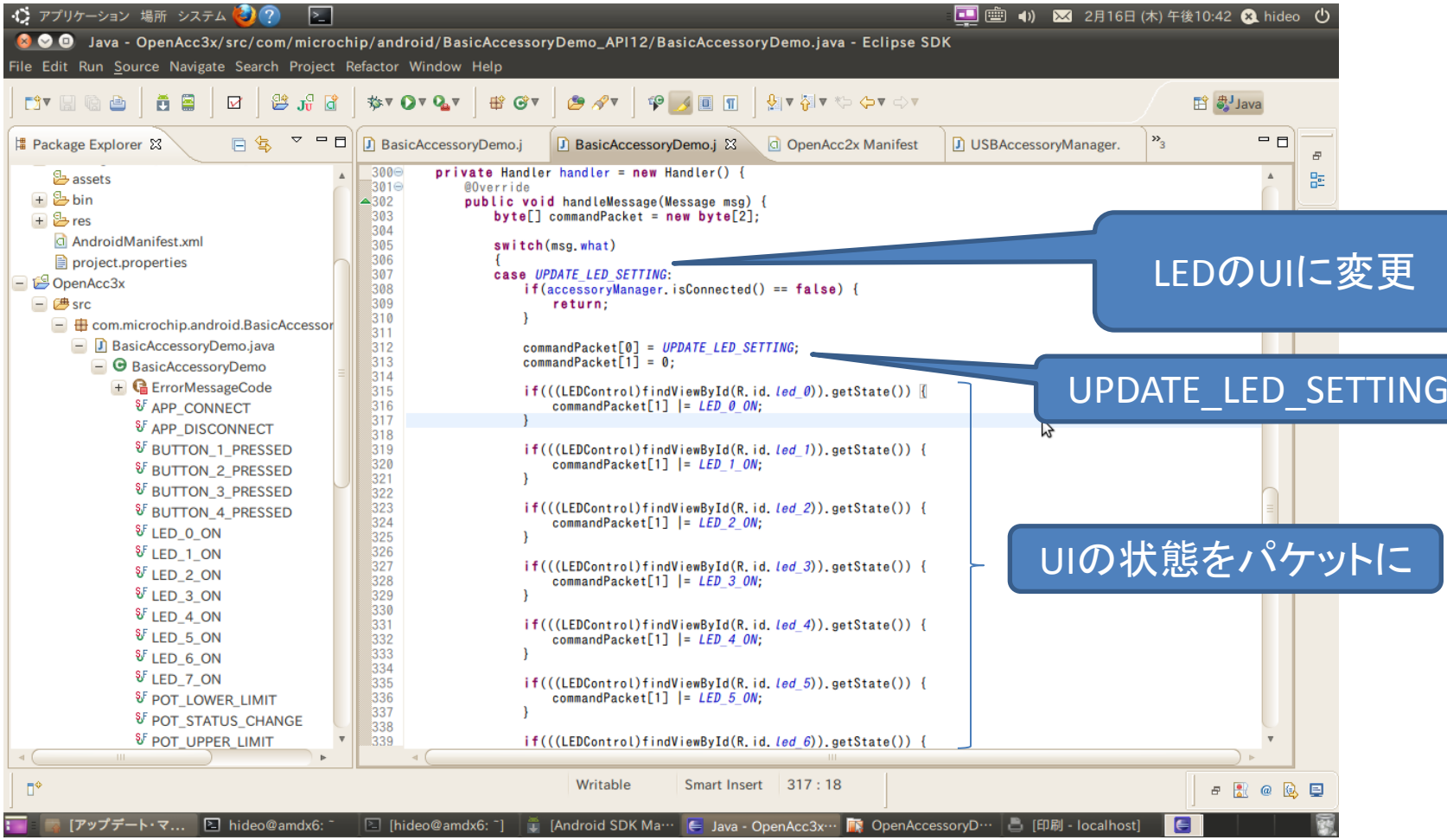
USBAccessoryManager.disable

- outputStreamをclose

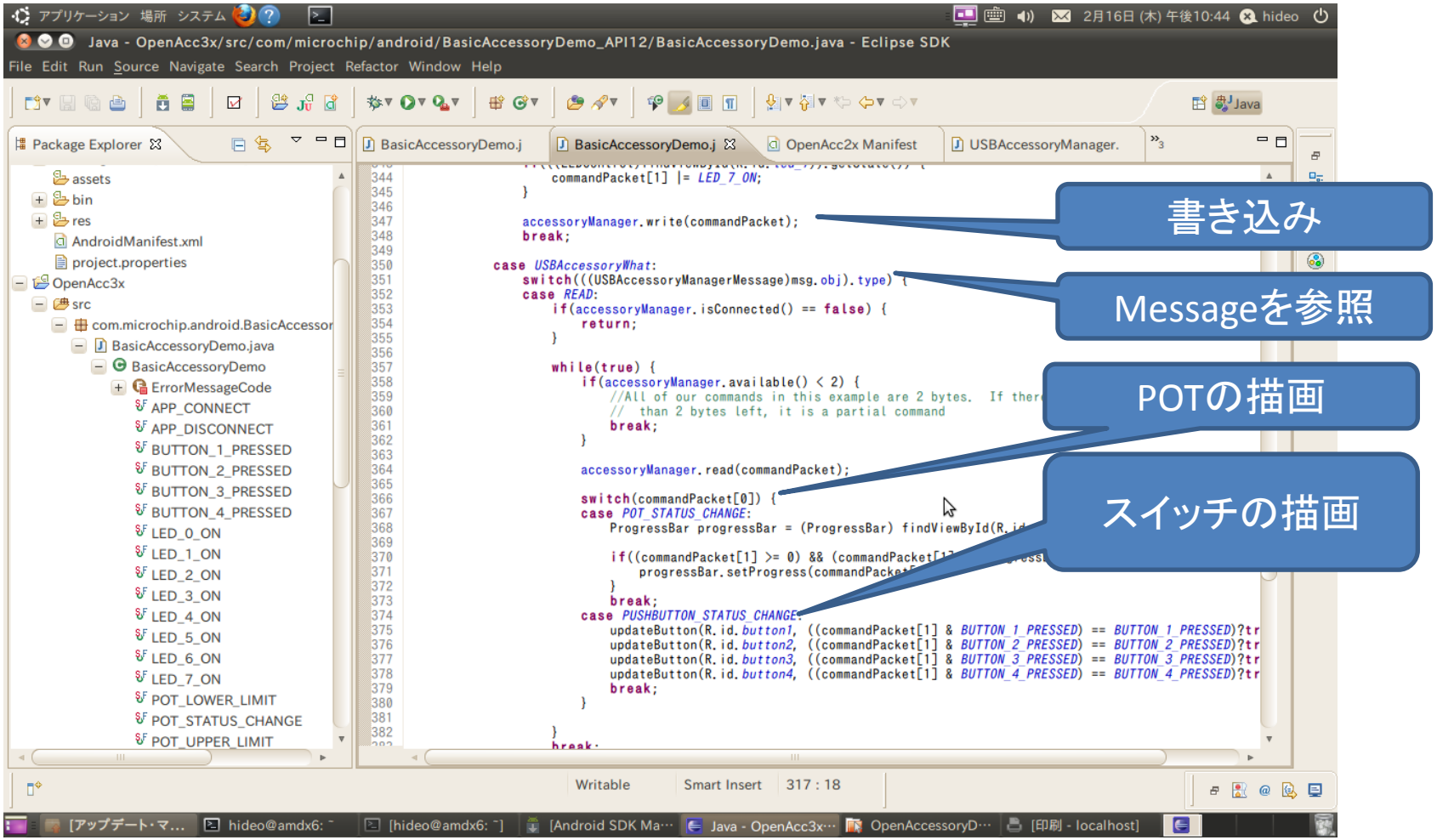
  733: outputStream.close();

BasicAccessoryDemo.java

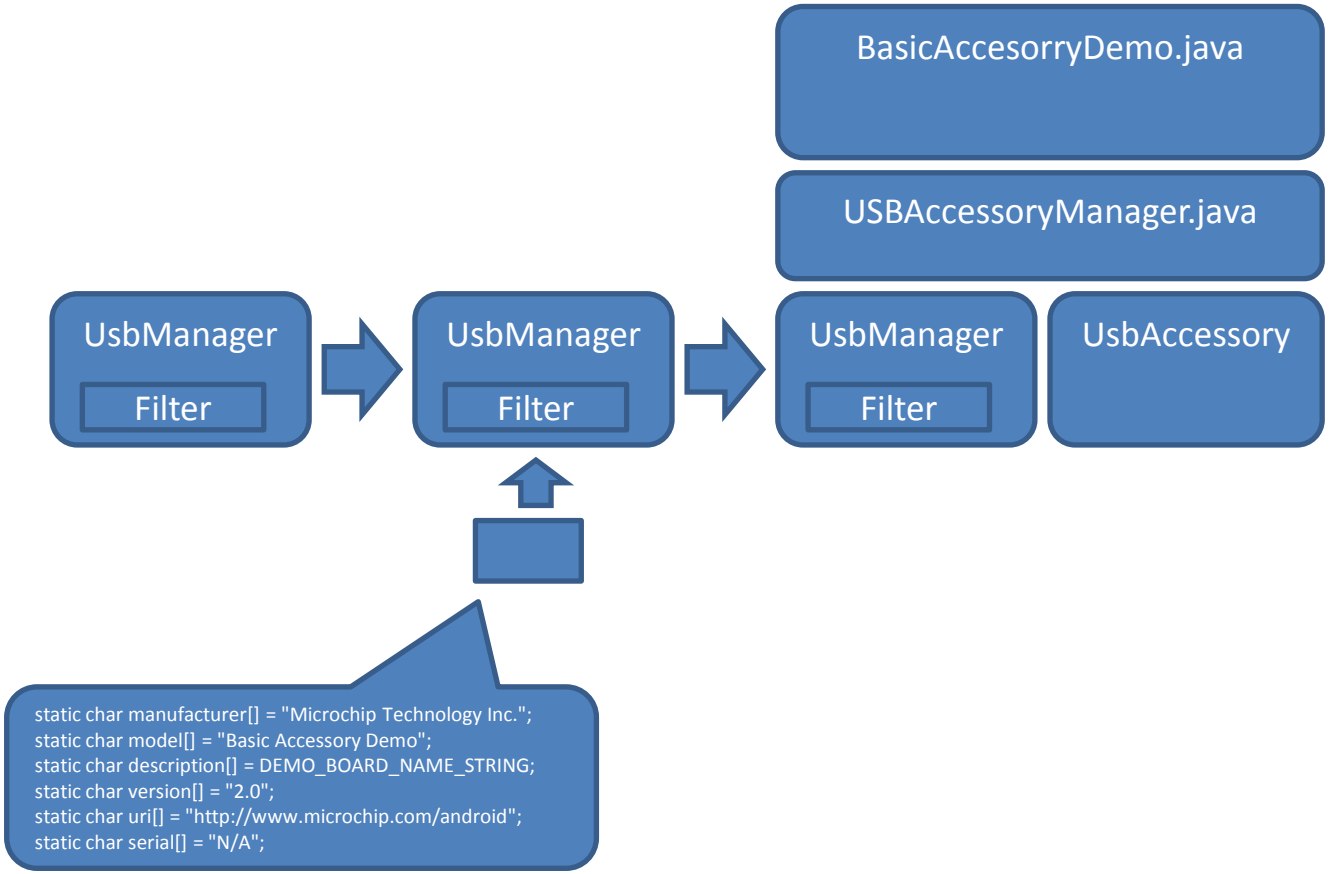
Handler(1)


Java - OpenAcc3x/src/com/microchip/android/BasicAccessoryDemo_API12/BasicAccessoryDemo.java - Eclipse SDK
File Edit Run Source Navigate Search Project Refactor Window Help
Package Explorer
BasicAccessoryDemo.j
BasicAccessoryDemo.j
OpenAcc2x Manifest
USBAccessoryManager.
assets
bin
res
AndroidManifest.xml
project.properties
OpenAcc3x
src
com.microchip.android.BasicAccessor
BasicAccessoryDemo.java
BasicAccessoryDemo
ErrorMessageCode
APP_CONNECT
APP_DISCONNECT
BUTTON_1_PRESSED
BUTTON_2_PRESSED
BUTTON_3_PRESSED
BUTTON_4_PRESSED
LED_0_ON
LED_1_ON
LED_2_ON
LED_3_ON
LED_4_ON
LED_5_ON
LED_6_ON
LED_7_ON
POT_LOWER_LIMIT
POT_STATUS_CHANGE
POT_UPPER_LIMIT
private Handler handler = new Handler() {
@Override
public void handleMessage(Message msg) {
byte[] commandPacket = new byte[2];
switch(msg.what)
{
case UPDATE_LED_SETTING:
if(accessoryManager.isConnected() == false) {
return;
}
commandPacket[0] = UPDATE_LED_SETTING;
commandPacket[1] = 0;
if(((LEDControl)findViewById(R.id.led_0)).getState()) {
commandPacket[1] |= LED_0_ON;
}
if(((LEDControl)findViewById(R.id.led_1)).getState()) {
commandPacket[1] |= LED_1_ON;
}
if(((LEDControl)findViewById(R.id.led_2)).getState()) {
commandPacket[1] |= LED_2_ON;
}
if(((LEDControl)findViewById(R.id.led_3)).getState()) {
commandPacket[1] |= LED_3_ON;
}
if(((LEDControl)findViewById(R.id.led_4)).getState()) {
commandPacket[1] |= LED_4_ON;
}
if(((LEDControl)findViewById(R.id.led_5)).getState()) {
commandPacket[1] |= LED_5_ON;
}
if(((LEDControl)findViewById(R.id.led_6)).getState()) {
Writable
Smart Insert
317 : 18
LEDのUIに変更
UPDATE_LED_SETTING
UIの状態をパケットに

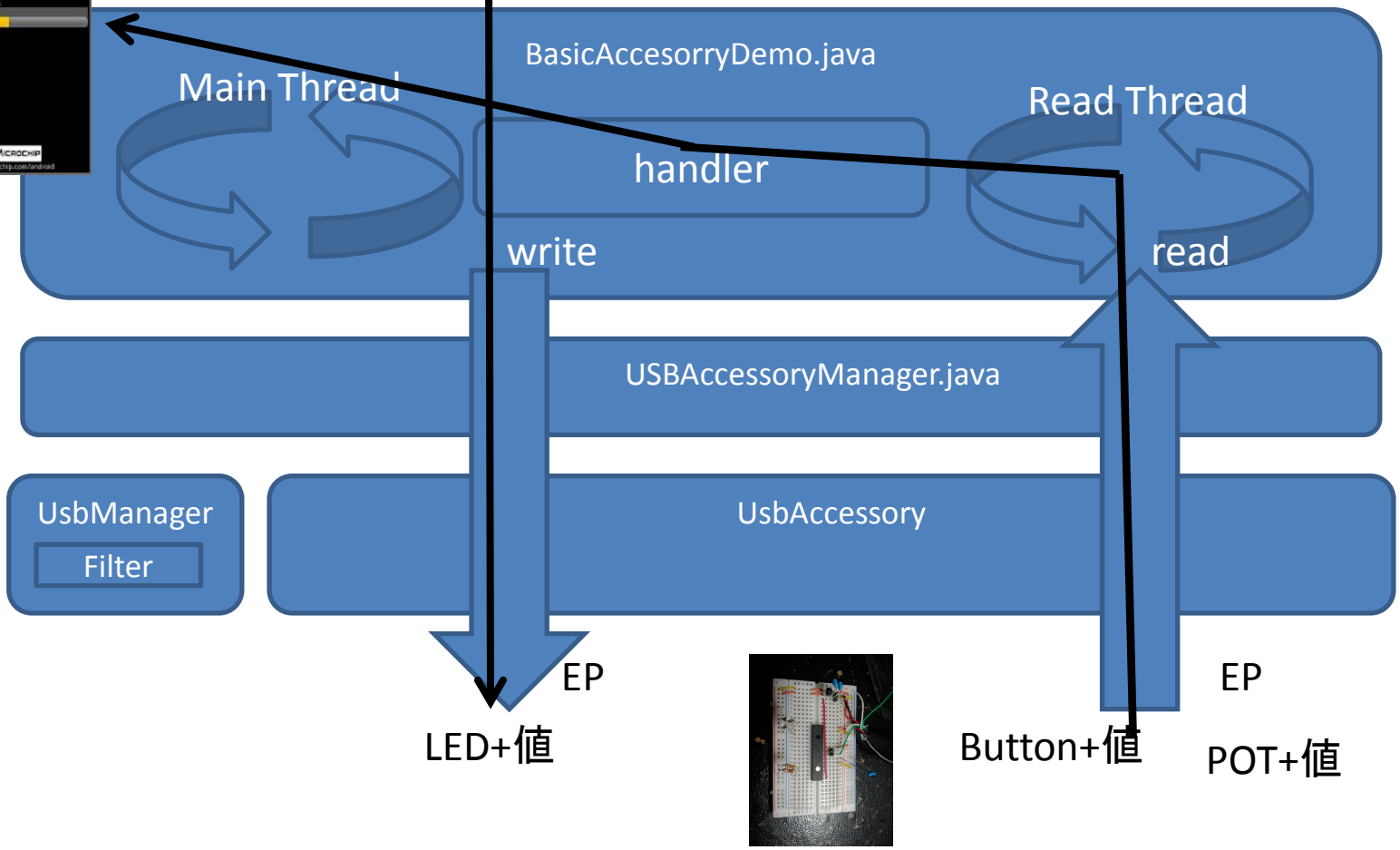
Handler(2)



APIの処理


BasicAccesorryDemo.java
USBAccessoryManager.java
UsbManager
Filter
UsbManager
Filter
UsbManager
Filter
UsbAccessory
static char manufacturer[] = "Microchip Technology Inc.";
static char model[] = "Basic Accessory Demo";
static char description[] = DEMO_BOARD_NAME_STRING;
static char version[] = "2.0";
static char uri[] = "http://www.microchip.com/android";
static char serial[] = "N/A";

プログラム構造


BasicAccesorryDemo.java
Main Thread
Read Thread
handler
write
read
USBAccessoryManager.java
UsbManager
Filter
UsbAccessory
EP
LED+値
Button+値
EP
POT+値

まとめ

- USB、ADKプロトコルを理解しなくても問題ない
  - Microchip社提供のソースに記述済
- プログラム変更に必要なこと
  - PICのピン・機能の割り当ての検討
  - アプリとのプロトコル{コマンド:値}を規定
  - 出力、入力、電圧値の入力の追加・削除
- このマイコンで他にできる事
  - PWM
    - モータ制御、LED輝度制御、RGB-LEDの色管理
  - SPI/I2C
    - 各種デジタルセンサ、周辺回路（CAN, RTC, 外部メモリなど）
  - UART
    - シリアル通信

おわり