

1-4 使う工具・道具一覧

① 必ず使う工具・道具



図1-7 必ず使う工具・道具

- [1] ハンダごて台 [2] ハンダごて※ [3] ハンダ線
 [4] 両面テープ [5] マスキングテープ
 [6] ビニールテープ [7] ワイヤストリッパ(カッター、ニッパで代用可能)
 [8] 圧着ペンチ(ラジオペンチで代用可能)
 [9] プラスドライバ [10] ニッパ [11] ラジオペンチ

② あると便利な道具・工具

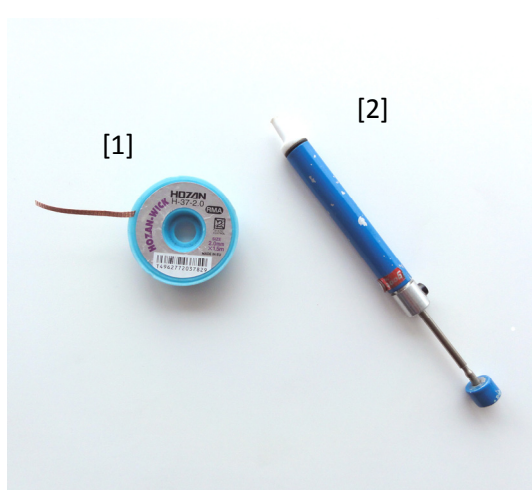


図1-8 あると便利な道具・工具

- [1] ハンダ吸い取り線
 [2] ハンダ吸い取り器
 ハンダ付けに失敗したら、ハンダ吸取り器やハンダ吸い取り線を使ってハンダを取り除き、修正します(*6)

※可能であれば30W以上のもの

(*6)
電動ハンダ吸い取機

本格的に吸い取ろうと思ったらコレ！
 盛りすぎた半田、間違えた半田を、電動パワーでガンガンと吸い取ることができます。



注:ここに挙げた工具群は秋葉原ラジオデパート、12章で紹介する千石電商、西川電子等で購入可能です。

2章 I/Oボードとマイコンボードを組み立てよう

概要

私たちが普段使っている、ほぼ全ての電化製品やパソコンの中に入っている電子回路基板は、製品の性能を決める重要な役割を担っています。そこに搭載されている電子部品を理解できるようになると、自分でオリジナルの回路を組み立てることもできます。(*7)

ここではI/O (アイ・オー) ボードの組み立てからはじめます。I/Oについては後ほどしっかり説明しますのでご心配なく。

- 2-1 ハンダ付けの基礎
- 2-2 シルク印刷と電子部品
- 2-3 I/Oボード作成
- 2-4 マイコンボード作成
- 2-5 「ロボット職人への技」 電源コネクタ作成
- 付録 コンデンサと抵抗器

2-1 ハンダ付けの基礎

ハンダ付けとは、(1)ハンダ(*8)によって電子部品を基板に接着させること、(2)基板の回路と電子部品との間に電気を通すことです。

ハンダは線状になっているのが一般的で、ハンダ線、糸ハンダ、あるいは単にハンダと言います。ハンダごてという、こて先が高温(*9)になる道具でハンダを溶かして接着します。

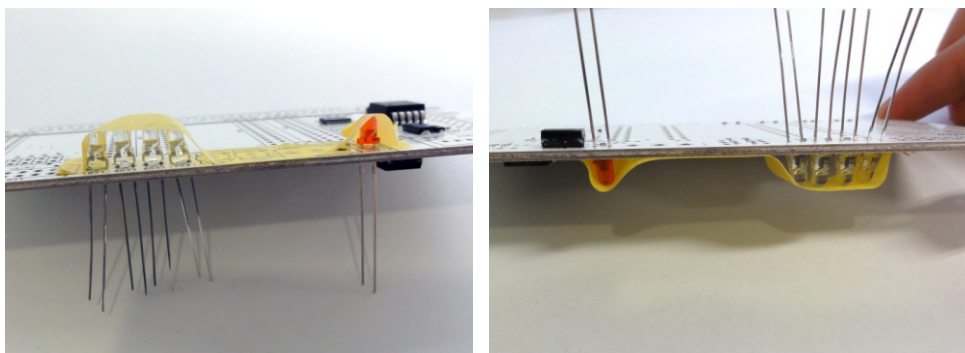


図2-1 ハンダとハンダごて

ハンダ付け手順

(1) 部品をキチンと固定する。(*10)

マスキングテープなどで基板と部品をしっかりと固定します。面倒でもしっかりと固定をすることで、部品が曲がったり、基板から浮いたりしてハンダ付けされてしまうのを防ぐことができます。



(*7)

分解のすすめ

いらなくなった電化製品をどんどん分解して中身を見てみましょう。センサ、LED、抵抗器などの電子部品の構成がわかるようになります。ただし、分解するとメーカー保証はなくなってしまうので注意！また、携帯電話は分解した後、再度組み立てて使用すると電波法違反になってしまうので気を付けましょう。

(*8)

ハンダの語源

錫(すず)と鉛が半々(50%ずつ)でできているからとも言われています。

(*9)

やけどに注意！

電源を入れるとハンダのこて先が高温になるので触らないように！ほとんどのハンダごてにスイッチはありません。



それでもやけどした場合、すぐに氷や水で冷やしましょう。

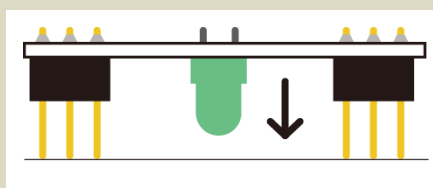
電源プラグをいれるといきなり熱くなるので、ハンダごて台に置いて熱くなるのを待ちましょう。



(*10)

【ポイント】ハンダ付けの順番

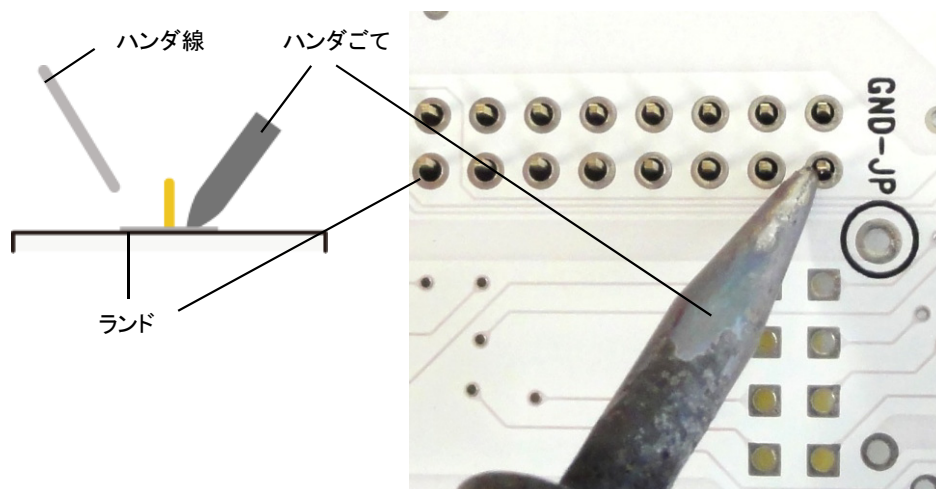
ハンダ付けは背の低い部品から順番に行いましょう。机で部品をおさえることができるので、部品を固定しながら作業できます。



背の高い部品から始めると、背の低い部品をつける際に部品が机から浮いてしまうので、曲がってついてしまったり、基板と部品の間に隙間があいてしまいます。

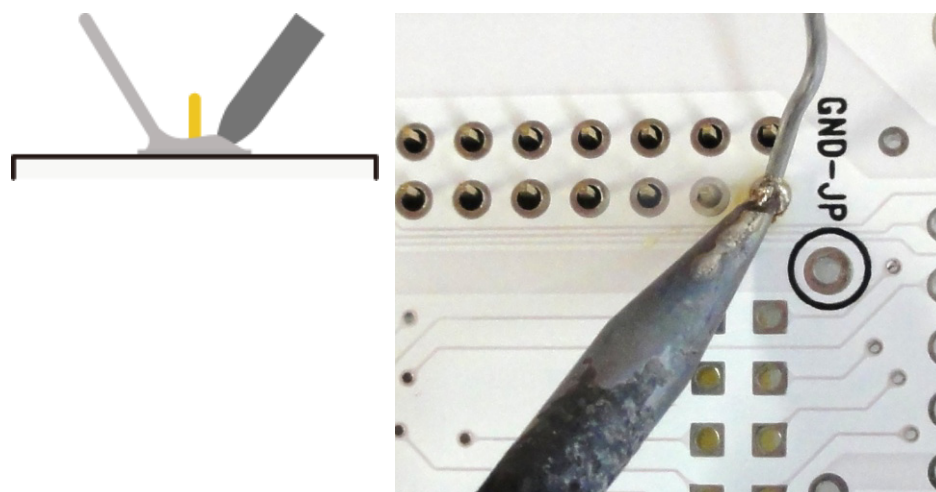
(2) ハンダごてをランドにあてて予熱

ランドとは銀色の輪っかの部分。実際にハンダ付けをするところです。
まずはこて先(*11)をランドの部分に当て、3秒ぐらい熱します。



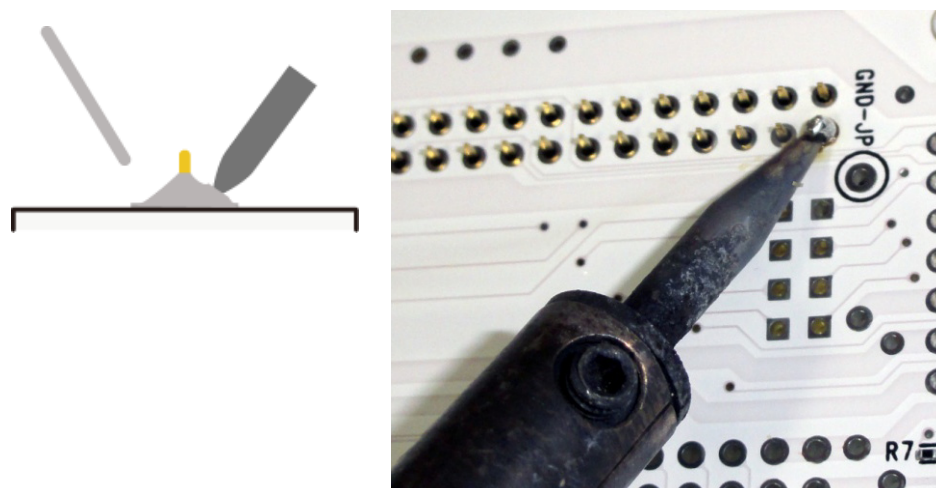
(3) ランドにハンダ線をつけてハンダを流す

こて先はそのまま、ランドにハンダ線をつけるとハンダが溶けてランドに流れ込みます。



(4) ハンダ線を離す

富士山のようなきれいなかたちになるように(*12)、ハンダごての熱でハンダを溶かします。



(*11)

【ポイント】こて先はいつもキレイに！

こて先が余分なハンダや焦げたハンダで汚れると、うまくハンダ付けができません。水でぬらして軽く絞ったスポンジで洗浄し、常に綺麗な状態にしておくことが大切です。



(*12)

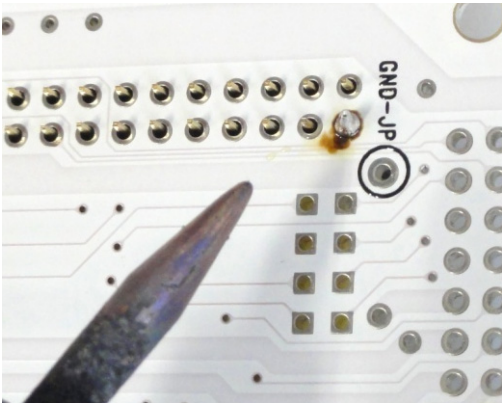
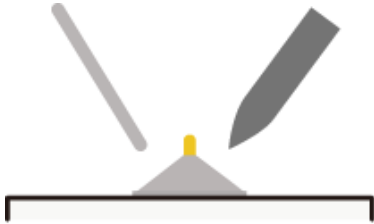
ハンダ付けの良し悪し

ハンダは多すぎても少なすぎてもいけません。球状に盛られたハンダは「いもハンダ」と呼ばれ、隣のランドとくっついて通電してしまう可能性があります。また少なすぎて部品の足にハンダがついていないものは「目玉ハンダ」と呼ばれます。均等に付いていないハンダはもろく、通電しなくなります。富士山型をめざして丁寧にハンダ付けしましょう。



いもハンダの例

- (5) ハンダごてを離す(*13)
ハンダが冷めて固まるのを待ちます。(*14)



2-2 シルク印刷と電子部品

基板には CN2, R2, LED3, SW3 など小さな文字が書いています。これはハンダ付けする部品の位置です。

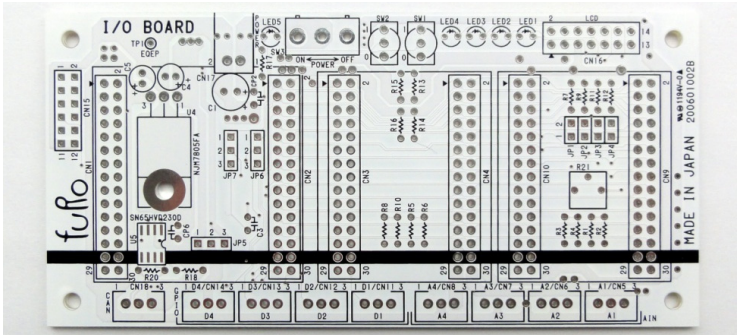


図2-2 基板のシルク印刷





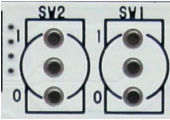
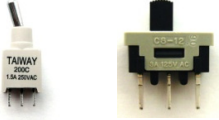
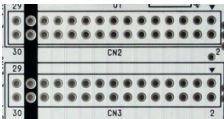

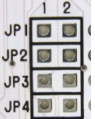
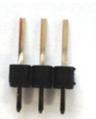
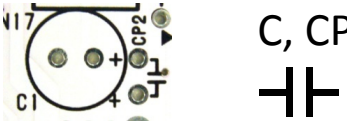
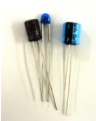
シルク印刷	対応する電子部品
	LED(発光ダイオード) 
	抵抗 
	スイッチ 
	コネクタ(またはピンヘッダ)(*15) 
	ジャンパピン (ピンヘッダ) 
	コンデンサ (キャパシタ) 

表2-1 シルク印刷

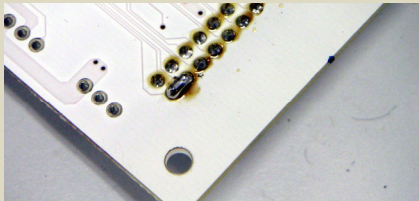
- (*13)
【ポイント】手順を守ろう

正しくハンダ付けをするためには、この(2)～(5)の手順の順番をしっかり守りましょう。

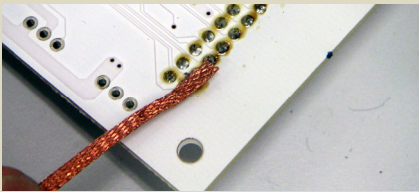
- (*14)
HOW TO 半田吸い取り

ハンダ付けに失敗したら・・・
半田吸い取り線を使ってみよう！

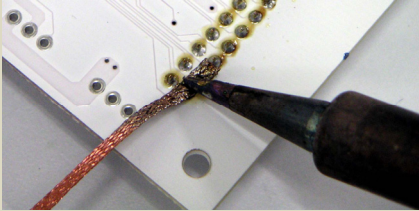
1. ヤバイ！盛りすぎた！



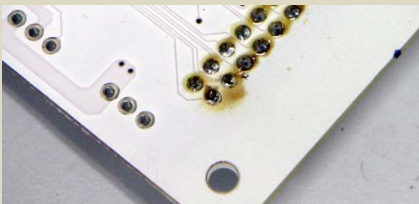
2. 半田線を吸い取りたい部位に乗せる



3. 吸い取り線の上から一緒に半田ごてで温めると・・・(ジュウジュウ)



4. この通り吸い取れました！



- (*15)
ピンヘッダとは

ピンヘッダは、隣のピン同士でショートさせるジャンパピンや、基盤同士を接続するコネクタピンなど、様々な用途に使われる便利なピンです。1列、2列などの種類があり、必要な長さに切って基板に実装します。

2-3 I/Oボード作成

I/OボードのI/O(アイ・オー)とは、INPUT(入力)/(出力)OUTPUTを略したものです。センサからの入力情報をマイコンに伝え、逆にマイコンからの指令(出力)をモータやLEDなどに伝える役割をもちます。人間でいうと目や耳から得られた情報を脳に伝え、脳からの指令を腕や足の筋肉に伝える「神経」にあたります。

準備するもの

部品(*16)

- | | | |
|---------------------|-----------------------|---------------------|
| (1) I/Oボード × 1 | (2) 1列ピンヘッダ × 1 (*17) | (3) 2列ピンヘッダ (2 × 4) |
| (4) 赤LED × 1 | (5) 透明LED × 4 | (6) 抵抗器 120Ω × 1 |
| (7) 抵抗器 510Ω × 1 | (8) 抵抗器 680Ω × 4 | (9) 抵抗器 1kΩ × 4 |
| (10) 抵抗器 2kΩ × 4 | (11) 抵抗器 3kΩ × 4 | (12) 抵抗器 10kΩ × 1 |
| (13) 15pF コンデンサ × 1 | (14) 0.1μF コンデンサ × 2 | |
| (15) 10μF コンデンサ × 2 | (16) 22μF コンデンサ × 1 | |
| (17) 三端子レギュレータ × 1 | (18) トグルスイッチ × 2 | |
| (19) スライドスイッチ × 1 | (20) 30ピンソケット × 4 | |
| (21) ジャンパ × 7 | (22) 電源コネクタピンヘッダ × 1 | |

道具

- ハンダ ●ハンダごて ●ニッパ ●マスキングテープ

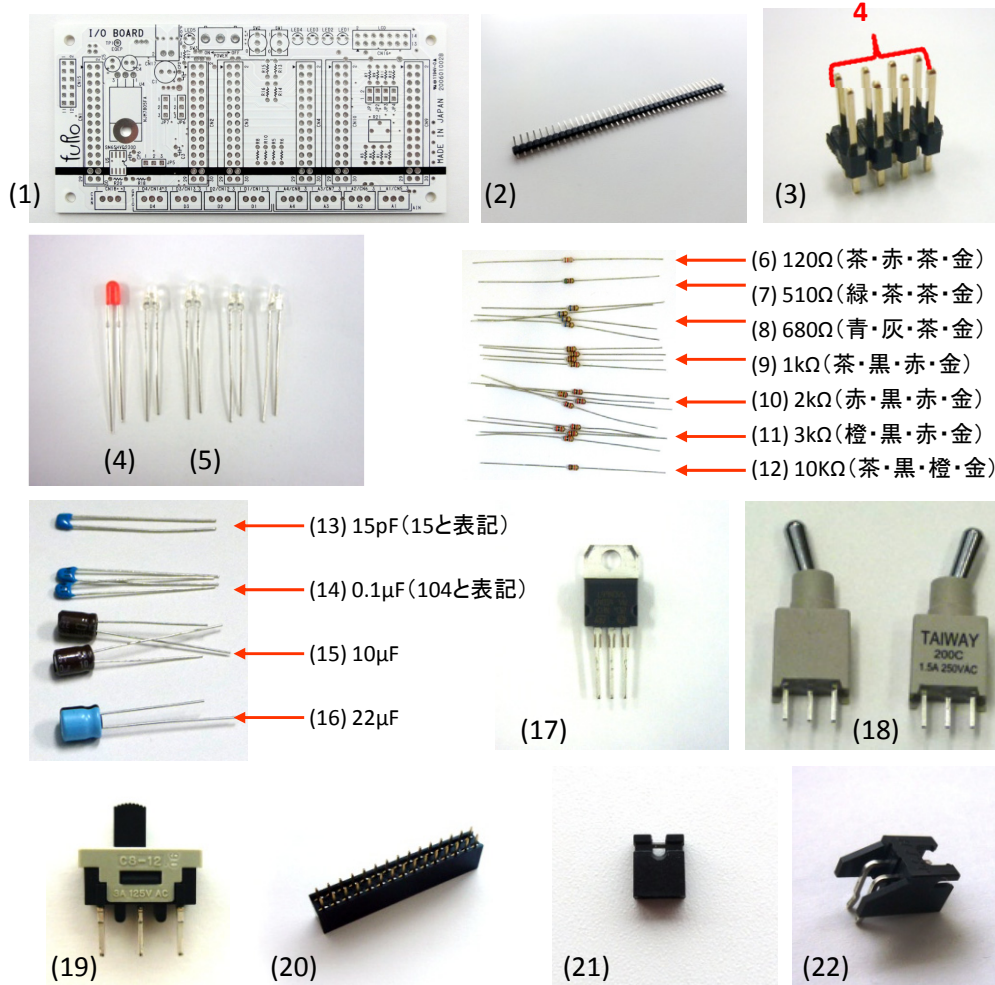


図2-3 I/Oボード作成用部品

(*16)

部品の入手先

ピンヘッダやLED、抵抗、コンデンサ、スイッチ、三端子レギュレータなどは秋葉原にある秋月電子通商やマルツパーツ館などで購入することができます(30ピンソケットはマルツパーツ館のみ)。それぞれ通販サイトもあるのでチェックしてみてください。

秋月電子通商

<http://akizukidenshi.com/>

マルツパーツ館

<http://www.marutsu.co.jp/>

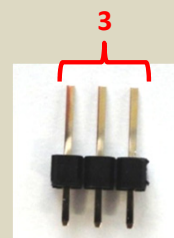
他にもRSコンポーネンツといったサイトでも電子部品を取り扱っています。

<http://jp.rs-online.com/>

(*17)

ピンヘッダのカット

1列ピンヘッダは、1×3ピンヘッダが12個できるようにカットしておきましょう。



ピンヘッダは、ニッパなどでカットします。切るときに勢いで切れ端が周囲に飛ばないように注意!



部品をハンダ付けする位置

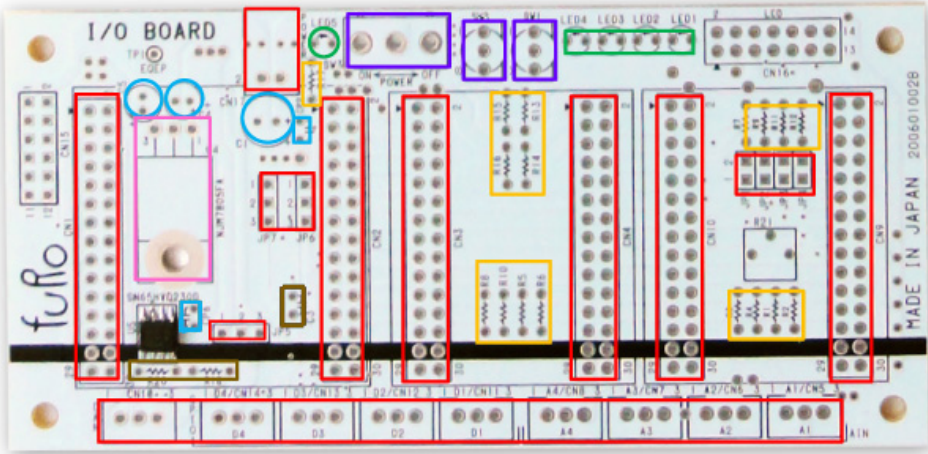


図2-4 部品をハンダ付けする位置

シルク印刷	ハンダ付けする部品
CN1, CN2, CN3, CN4, CN9, CN10	30ピンソケット
CN5, CN6, , CN7, CN8, CN11 , CN12 , CN13,CN14, , CN18, JP5, JP6, JP7	1×3ピンヘッダ
JP1, JP2, JP3, JP4	2×4ピンヘッダ
CN17	電源コネクタピンヘッダ
R1, R3, R5, R8	抵抗器 2kΩ (赤・黒・赤・金)
R2, R4, R6, R10	抵抗器 3kΩ (橙・黒・赤・金)
R7, R9, R11, R12	抵抗器 680Ω (青・灰・茶・金)
R13, R14, R15, R16	抵抗器 1kΩ (茶・黒・赤・金)
R17	抵抗器 510Ω (緑・茶・茶・金)
LED1, LED2, LED3, LED4	透明LED
LED5	赤LED
C1	22μF コンデンサ
CP2, CP6	0.1μF コンデンサ
C4, C5	10μF コンデンサ
SW1,SW2	トグルスイッチ
SW3	スライドスイッチ
U4	三端子レギュレータ

CAN使用時

R20	抵抗器120Ω(茶・赤・茶・金)
R18	抵抗器10kΩ(茶・黒・橙・金)
C3	15PFコンデンサ

表2-2 シルク印刷と部品の対応表
(※抵抗器の一番右のカラー「金」は「銀」の場合もあります)

(*18)
SI接頭辞

1kΩの「k(キロ)」や、22μFの「μ(マイクロ)」をSI(国際単位系)接頭辞といい、メートル(m)やアンペア(A)といった国際単位系(SI)の前に付いて、大きな量や小さな量を表します。

例えば「k(キロ)」は10の3乗を示す接頭辞なので、このようになります。

1kΩ=1×10^3Ω=1000Ω

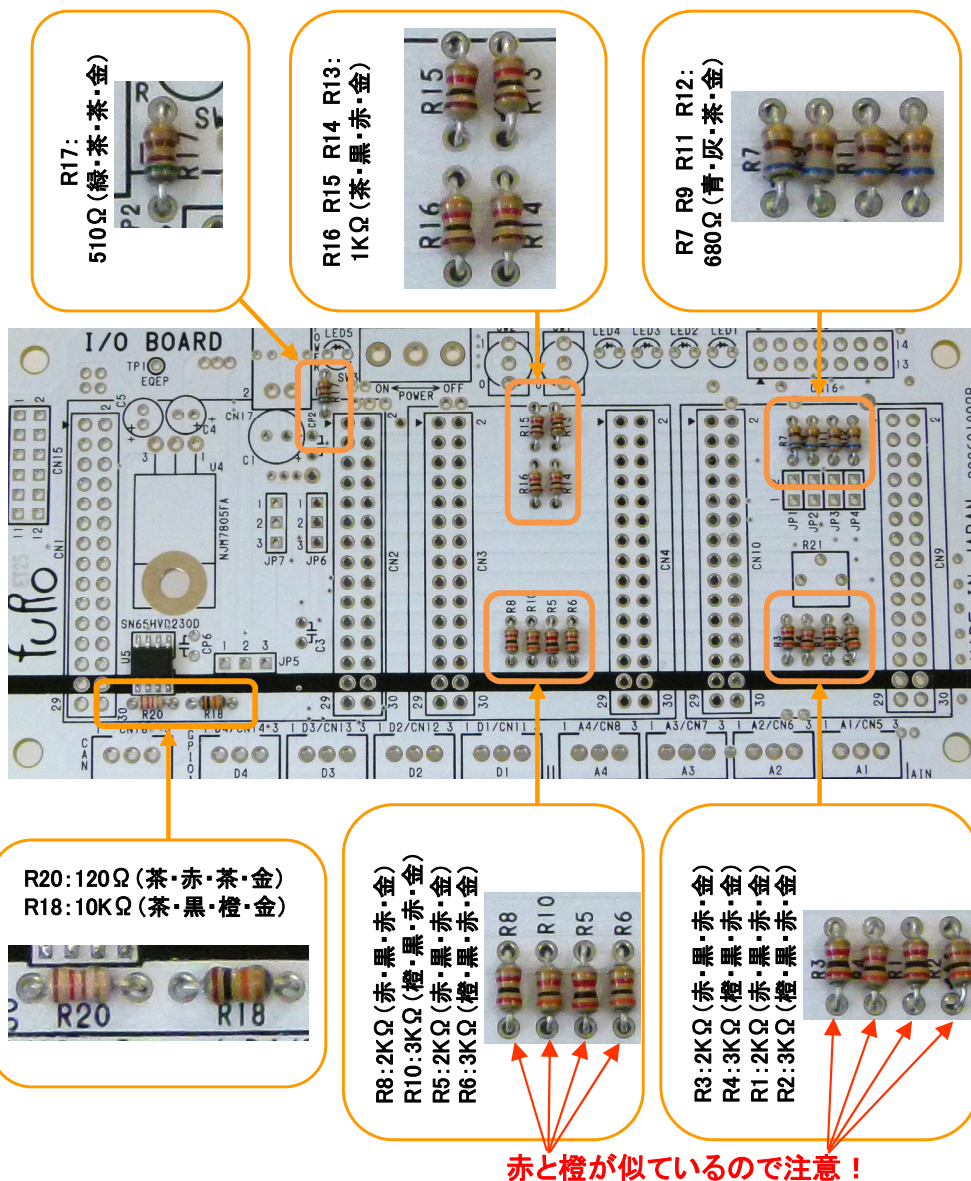
SI接頭辞一覧

10^12	T(テラ)
10^9	G(ギガ)
10^6	M(メガ)
10^3	k(キロ)
10^2	h(ヘクト)
10^1	da(デカ)
10^0	なし
10^-1	d(デシ)
10^-2	c(センチ)
10^-3	m(ミリ)
10^-6	μ(マイクロ)
10^-9	n(ナノ)
10^-12	p(ピコ)

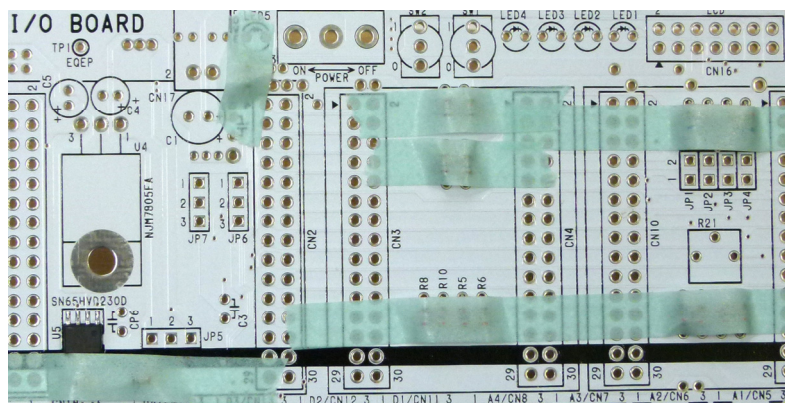
2-1のハンダ付けの基礎で説明した通り、背の低い部品から順番にハンダ付けをしていきます。

① 抵抗器のハンダ付け

抵抗器(*19)は全部で19個ハンダ付けします。それぞれの抵抗の値(*20)や取り付け場所を間違わないように、よく確認しましょう。



抵抗器を差し込んで(*21)、抵抗値と取り付け場所を再度確認したら、表面からマスキングテープで抵抗器を固定します。



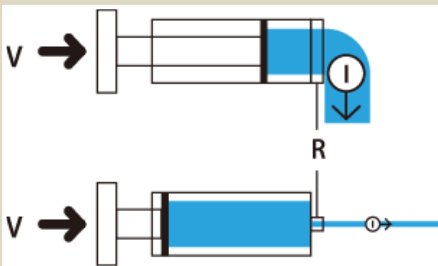
(*19) オームの法則

抵抗(resistance)はRで、抵抗値はΩで表します。オームの法則は、電圧、抵抗、電流の関係を示したもので、

$$V=R \times I \quad (\text{電圧}=\text{抵抗} \times \text{電流})$$

VRI(ブリちゃん)と覚えましょう。ただし、VがEと表記されていることもあるので注意。

これらの関係は水鉄砲で表すことができます。水鉄砲を押す力を電圧(V)、水鉄砲の口を抵抗(R)、水の流れを電流(I)とします。水鉄砲の口を小さくする(抵抗器を付ける)と、水は細く永く流し続けることができます。



【豆知識】

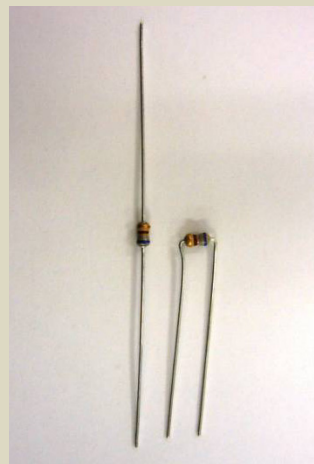
ドイツの物理学者ゲオルク・オームによって公表されたため、「オームの法則」と名付けられました。でも発見したのはヘンリー・キャヴェンディッシュという別の人です。

(*20) 抵抗器のカラーコードの読み方

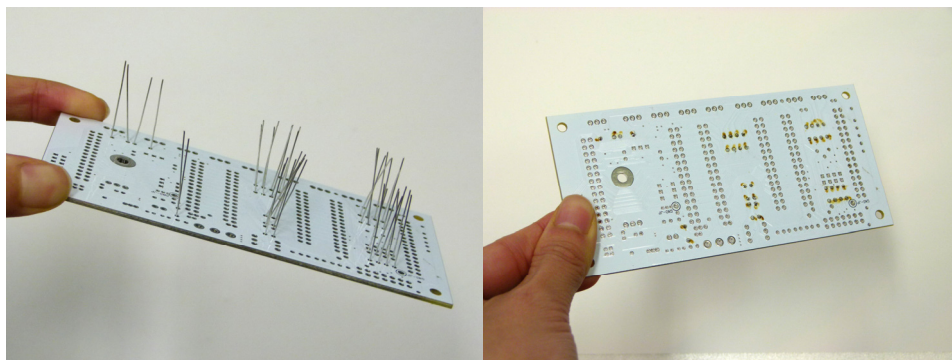
P22「付録 コンデンサと抵抗器」を参照のこと。

(*21) 抵抗器の準備

抵抗器は足を曲げて、ハンダ付け用の穴に差し込みます。



抵抗器を固定したらボードを引っくり返し、ハンダ付けします。(*22)

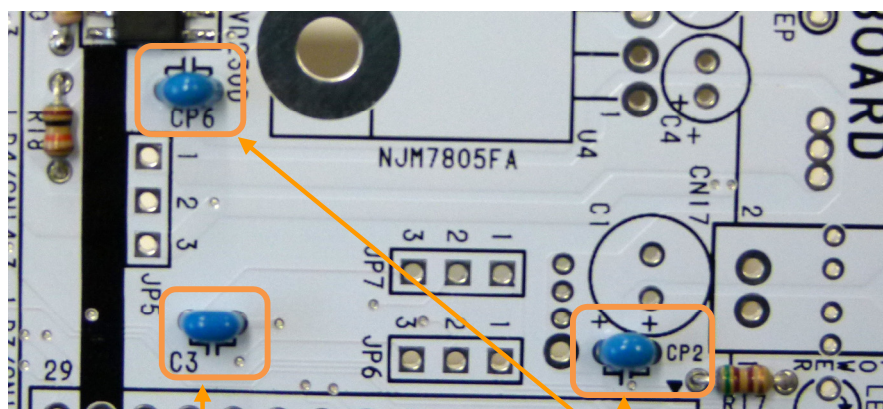


ハンダ付けした後、飛び出している抵抗器の足をカットします。

他の部品のハンダ付けも、基本は抵抗器と同じく、「取り付け場所を確認して差し込む」、「マスキングテープで固定」、「ハンダ付けする」、「長い足の場合はカットする」の手順で行います。

② CP2、CP6、C3にコンデンサをハンダ付け

CP2とCP6に0.1 μ Fのコンデンサ(青色部分に「104」と書かれた小さいコンデンサ)(*23)、C3に15pFのコンデンサ(同様に「15」と書かれたコンデンサ)を差し込みます。



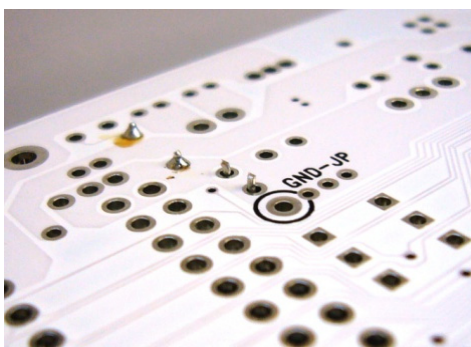
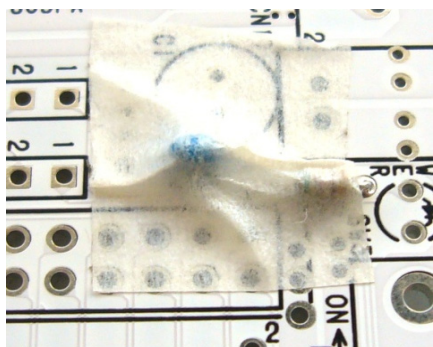
C3: 15pF (15と表記)



CP2、CP6: 0.1 μ F (104と表記)

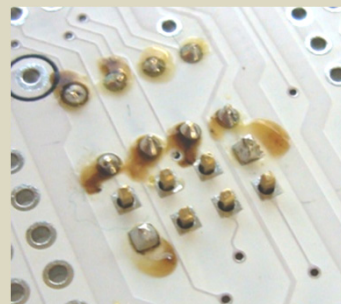


抵抗器と同じように表側からマスキングテープで固定し、裏返してハンダ付けします。ハンダ付けしたら長い足をカットします。



(*22) ボードが焦げた！？

ハンダに含まれているヤニ(フラックス)が、ハンダ付けの際に焦げたり茶色に変色したりしたものです。特に、I/Oボードは白色なので目立ちますが、機能には問題ありません。



どうしても気になる！という人はヤニ(フラックス)除去剤を用いると、きれいになります。『フラックスクリーナー』等の名前で販売されています。



(*23) セラミックコンデンサ

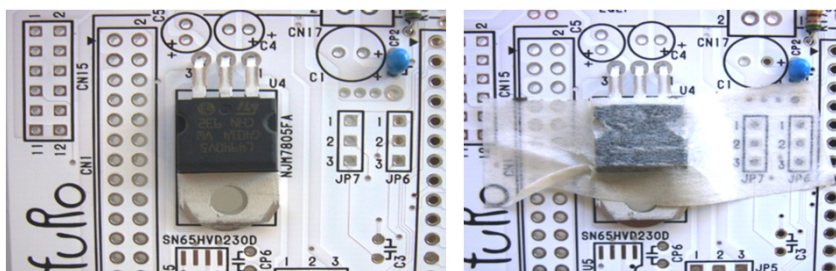
0.1 μ F、15pFのコンデンサに使っているセラミックコンデンサには極性がなく、足の長さが同じ。どちらの向きに差し込んでも大丈夫です。



なお、コンデンサについてもっと知りたい人は、P22「付録 コンデンサと抵抗器」を参照のこと。

③ U4に三端子レギュレータをハンダ付け

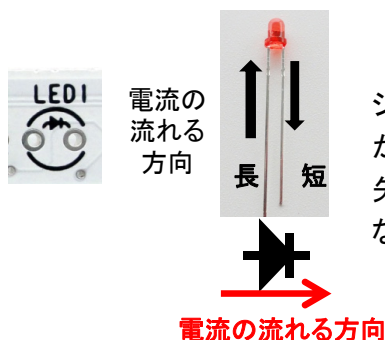
三端子レギュレータ(*24)の足を曲げ、ボードの穴と三端子レギュレータの穴を合わせて金属の面がI/Oボードの面と合うようにU4へ差し込み、マスキングテープで固定します。



固定したらボードを裏返し、ハンダ付けして足をカットします。

④ LEDのハンダ付け

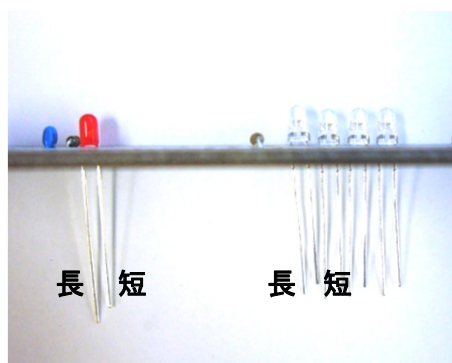
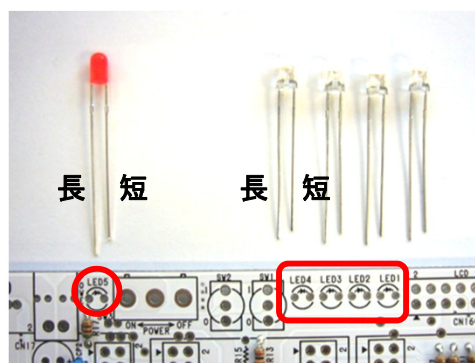
LED1~4に透明のLED(*25)を、LED5に赤色のLEDを、向きに注意して差し込みます。LEDは電流の流れる方向が決まっています、差し込む向きを間違えると動きません。



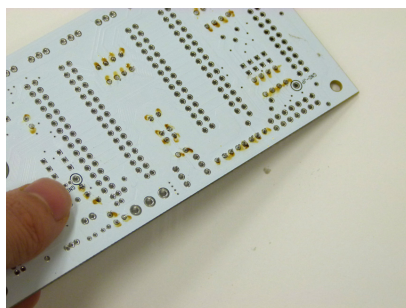
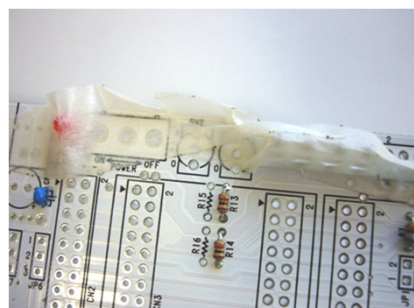
ATTENTION!!

シルク印刷の矢印の向きと、電流の方向が合うように差し込みます。長い方の足が矢印の根元に、短い方の足が矢印の先になります。

差し込んだら横方向からも見て、しっかり確認してください。



確認が済んだらマスキングテープで固定をし、ボードを裏返してハンダ付けして足をカットします(*26)。



(*24)

三端子レギュレータとは

「レギュレータ」は電源部に使われる電子部品で、ある入力電圧に対して、それ下の任意の一定電圧を出力します。三つの足(端子)が出ているので、三端子レギュレータと呼ばれます。

三端子レギュレータは、足を写真のようにラジオペンチで90度曲げてボードに差し込みます。



(*25)

LED(発光ダイオード)とは

LEDは電流が流れると発光する電子部品で、電光掲示板などの表示器やマイコンの動作確認などに使われます。I/Oボードでは、プログラムの動作状態をモニターする用途などに使用しています。

LEDには、様々な色や大きさ、形のものがあります。また、電流の流し方や大きさによって、光る色が変わるものもあります。



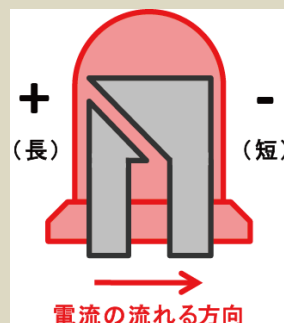
マイコンボード表面に搭載されている、この小さな四角い部品もLED!



(*26)

LEDに電流が流れる方向

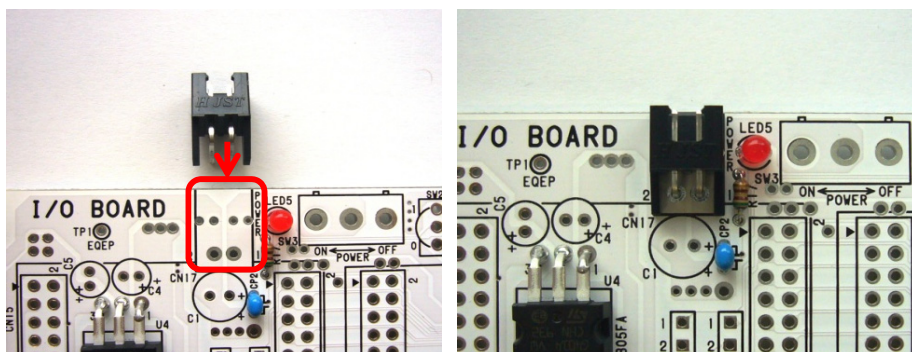
LEDの足をカットした後、向きが分からなくなっちゃった!!



中の金属を見て、上側の大きい金属がついている方がカソードです。

⑤ 電源コネクタピンヘッダをハンダ付け

LED5の左隣にあるCN17(POWER)に電源コネクタピンヘッダを、カチッとハマるまで差し込みます。

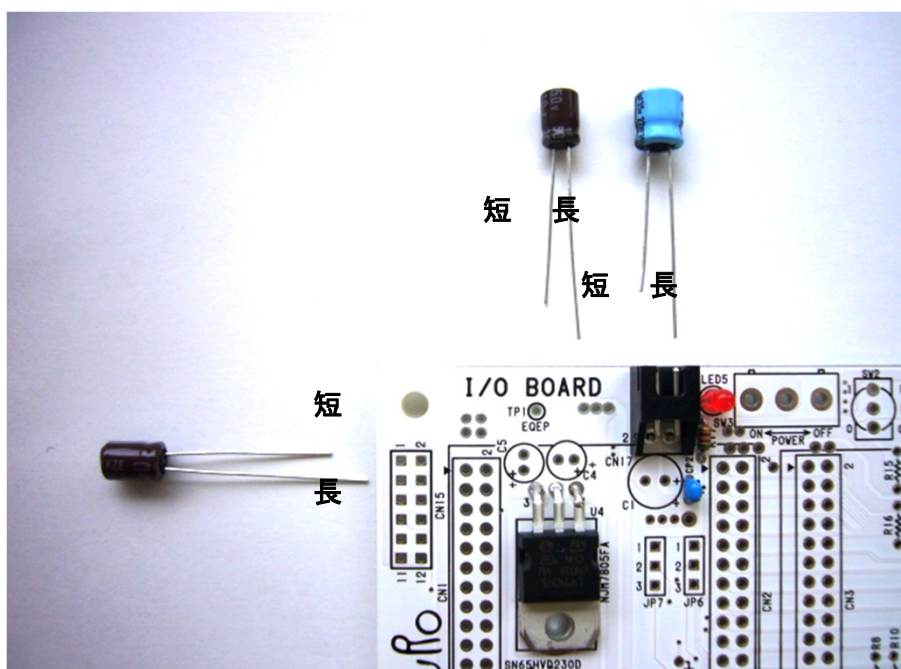


裏返してハンダ付けします。(*27)



⑥ 10μFコンデンサと22μFコンデンサのハンダ付け

22μFコンデンサ(水色の太めの方)をC1に、10μFコンデンサ(チョコレート色の方)をC4とC5に、足の長い方が「+」の表示がある穴に入るように差し込みます。(*28)



(*27)

電源コネクタ実装時の注意



ATTENTION!!

電源コネクタの2つの端子のハンダがくっつかないように気をつけましょう。接触していると、電源の+と-が短絡(ショート)してしまい、マイコンやI/Oボードが**一発で壊れて**しまいます。

(*28)

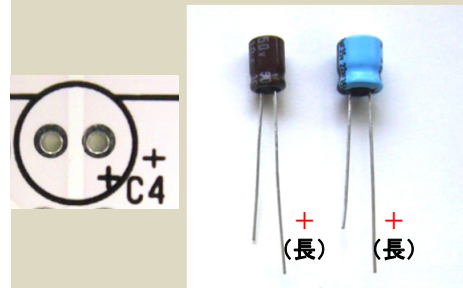
電解コンデンサを差す向き



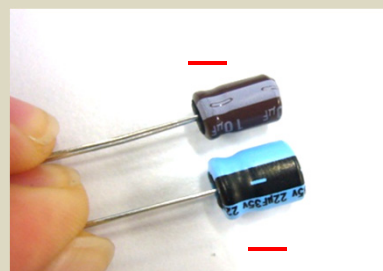
ATTENTION!!

10μF、22μFコンデンサに使っている電解コンデンサには極性があり、間違えると**破裂**する恐れがあります。しっかり確認しましょう。

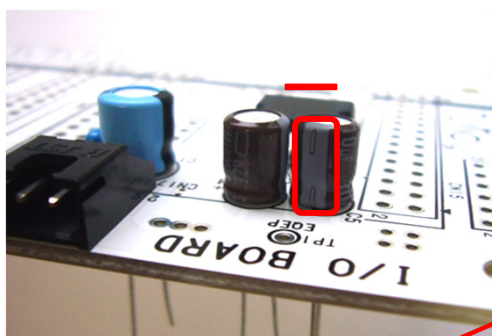
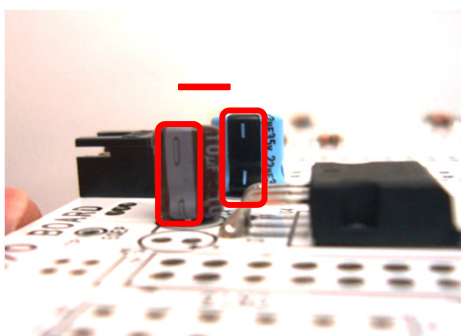
足の長い方が「+」です。シルク印刷の+とコンデンサの+が合うように差します。



マイナス側の側面には「-」の記号がプリントされています。



三端子レギュレータ側から見てC1とC5のコンデンサのマイナスのプリントが、電源コネクタピンヘッダ側から見てC4のコンデンサのマイナスのプリントが見えるかどうか確認してください。



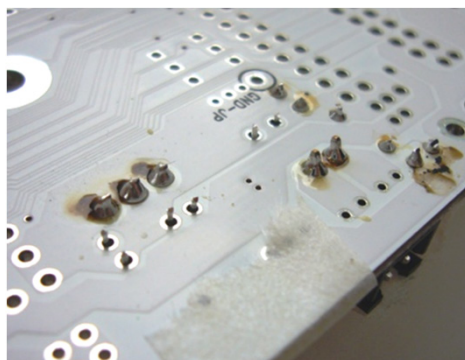
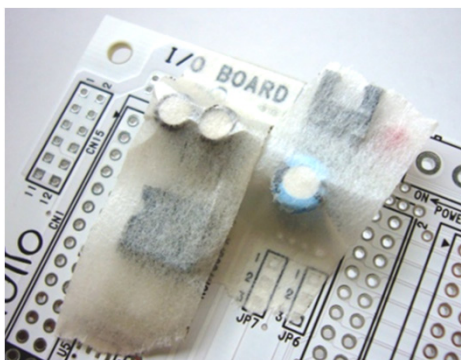
向きを間違えるとコンデンサが破裂します！(*29)
必ず何度も確認してください！

(*29)
本当に爆発します。

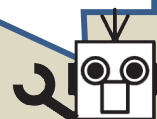
【WARNING!! 良い子はマネしない】
コンデンサー逆差し実験
木端微塵に爆発しました！！(XoX)



確認が済んだらマスキングテープで固定をし、ボードを裏返してハンダ付けし、足をカットします。



本当に気を付けないと
爆発するヨ

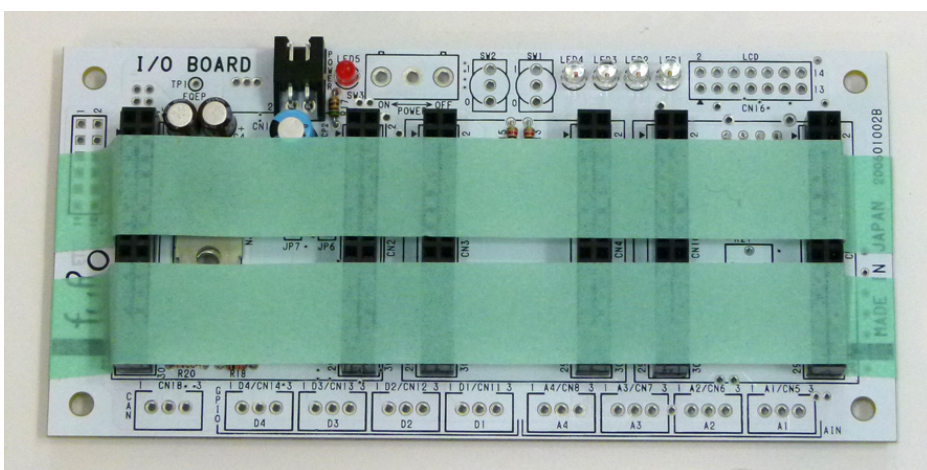


(*30)
【ポイント】コネクタのハンダ付けの順番

最初に部品の四隅をハンダ付けし、部品が浮いていないか確かめます。部品が浮いていたら、ランドを熱して、つけたハンダを溶かしながら修正します。

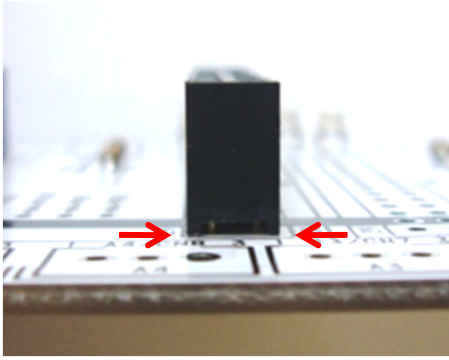
⑦ 30ピンソケットのハンダ付け

CN1、CN2、CN3、CN4、CN9、CN10に30ピンソケットを表側から差し込み、マスキングテープで固定します。

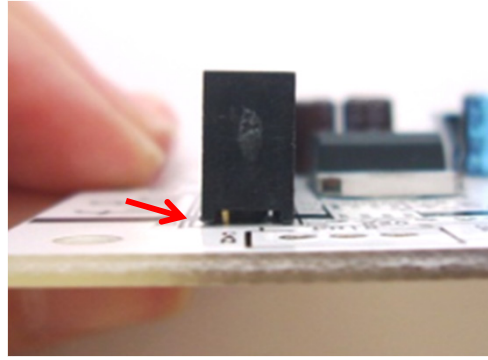


固定したらボードを裏返し、まず四隅をハンダ付けし、もう一度表側にして、ピンソケットがボードに対して曲がったり浮いたりしていないかを両側面から確認します(*30)。



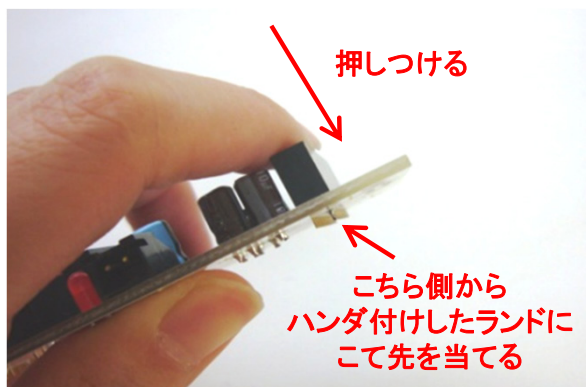


まっすぐハンダ付けされている
(両側とも下部がボードに接地)



曲がってハンダ付けされている
(左側が浮いています)

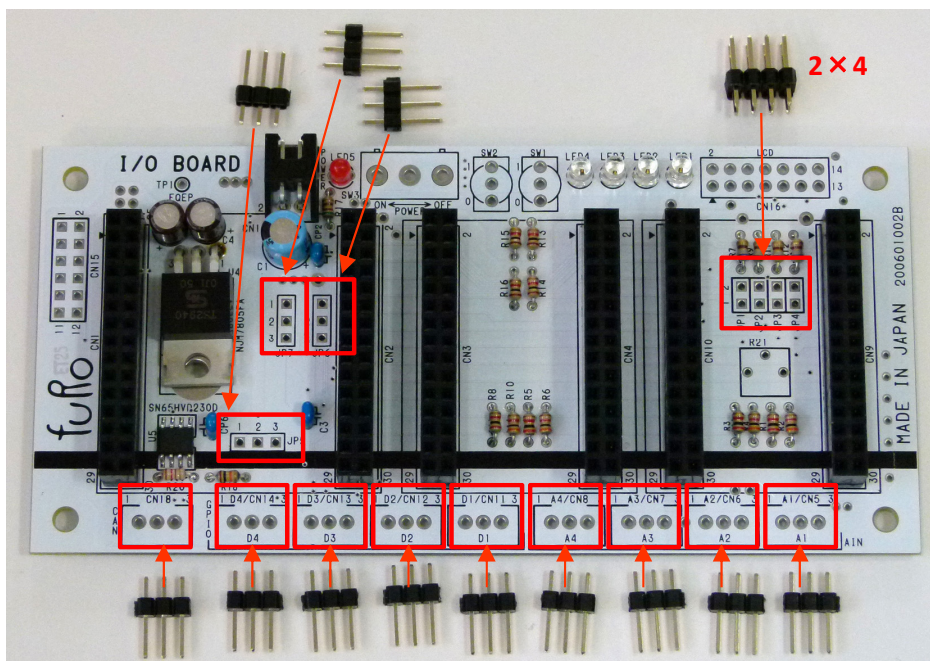
曲がったり浮いたりしていた場合は、手や机の面でピンソケットをボードにしっかり押しつけながら、ハンダ付けしたランドをハンダごてで温めて修正します。



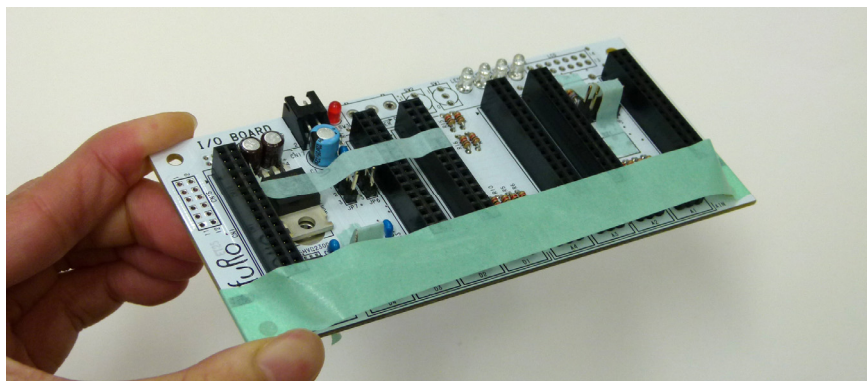
まっすぐハンダ付けされていることが確認できたら、四隅以外の残りの部分もハンダ付けします。

⑧ ピンヘッダのハンダ付け

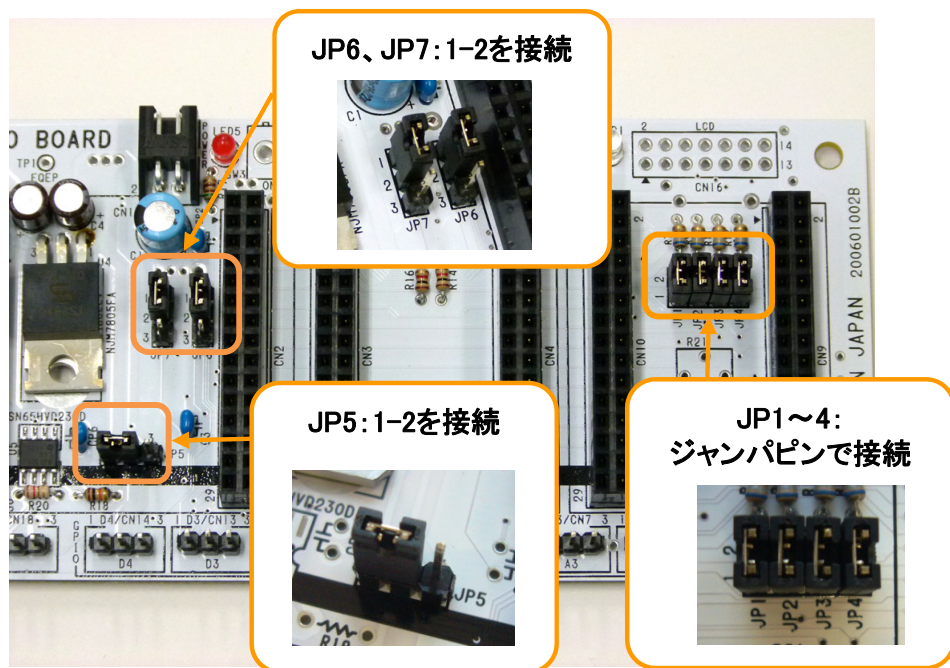
JP1-4に2×4ピンヘッダを、CN5、CN6、CN11、CN14、JP5、JP6に1列ピンヘッダからカットした1×3ピンヘッダを差し込みます。



差し込んだらマスキングテープで固定し、裏返してハンダ付けします。(*31)

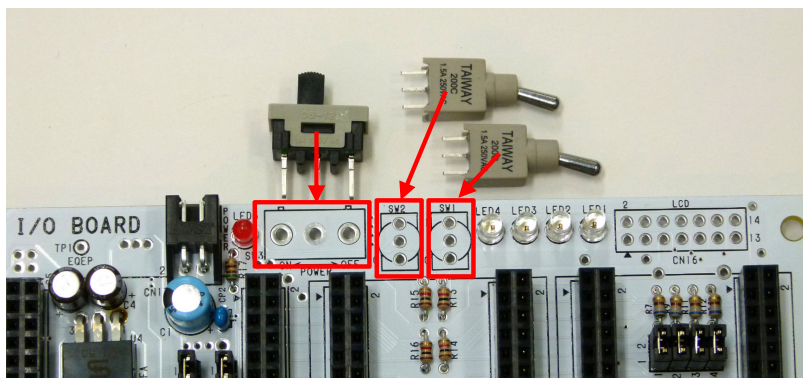


ハンダ付けが終わったら、JP1-4とJP5、JP6、JP7の1と2が接続されるようにジャンパピンを差し込みます。



⑨ スライドスイッチ、トグルスイッチのハンダ付け

SW3にスライドスイッチ(*32)を差し込み、マスキングテープで固定してハンダ付けします。



同様に、SW1、SW2にトグルスイッチ差し込み、マスキングテープで固定してハンダ付けします。

(*31)

ピンヘッダが浮いていたら

ピンヘッダが浮いてハンダ付けされてしまった場合は、机の面などでピンヘッダを押さえながら、裏からハンダごてをランドに当てて修正します。手で押さえようとすると、ピンが熱くなるので危険です。

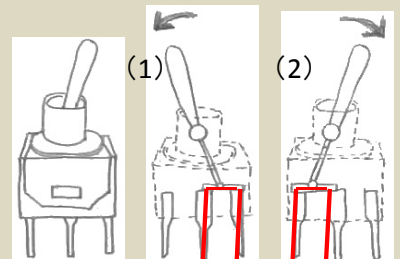
(*32)

スイッチとは

スイッチのON/OFFは、私たちも日常的に行っています。部屋の電気を点けたり消したり、目覚まし時計のアラームを止めたり。スイッチは電気回路を切ったりつないだりする役割を持っています。スイッチの形や回路のつなぎ方はスイッチによってそれぞれですが、ここではトグルスイッチとスライドスイッチの2種類を使います。

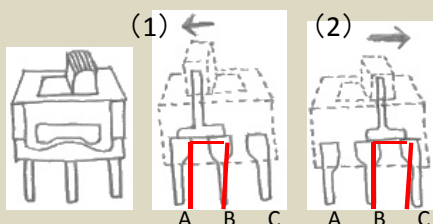
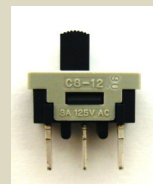
トグルスイッチはレバーを上下(取り付け方によっては左右)に倒して、3本ある足(端子)の内2つをつなげて通電させるスイッチ。レバーを倒した側と反対側の端子と中央の端子が導通します。

トグルスイッチとその仕組み



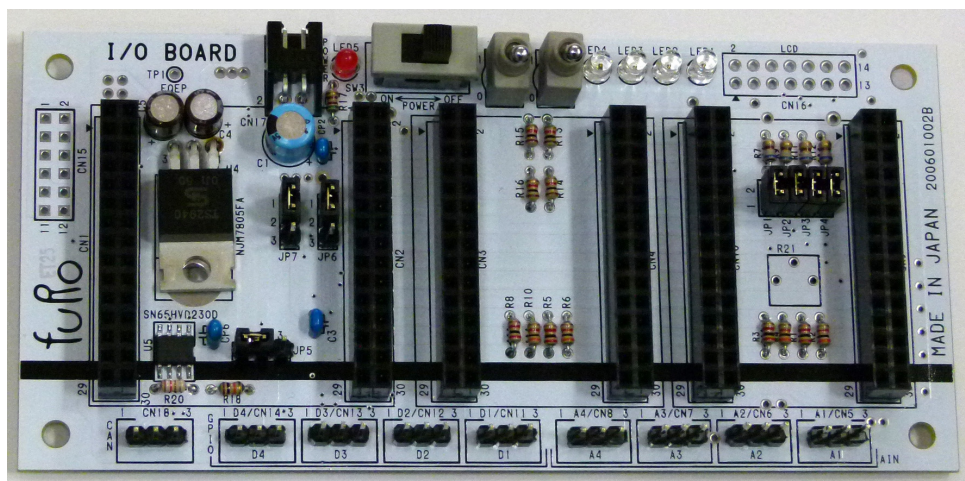
(1) BとCが通電 (2) AとBが通電

スライドスイッチとその仕組み



(1) AとBが通電 (2) BとCが通電

⑩ 完成! (*33)

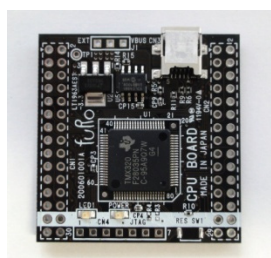


2-4 マイコンボード作成

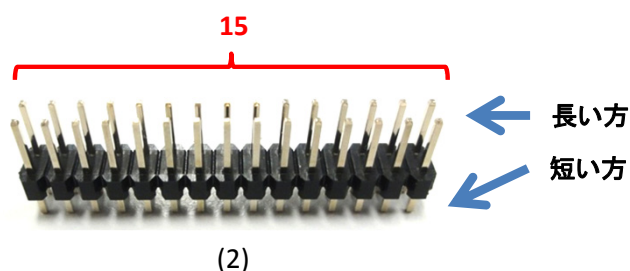
I/Oボードに比べて小型で、ハンダ付けする箇所も少ないのですが、ピンヘッダが曲がってついてしまったりすると、I/Oボードとうまく接続されなかったりします。丁寧に作りましょう。

準備するもの

部品 (1) マイコンボード×1 (2) 2列ピンヘッダ (*34)



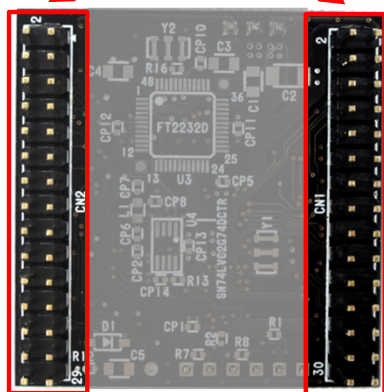
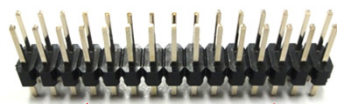
(1)



(2)

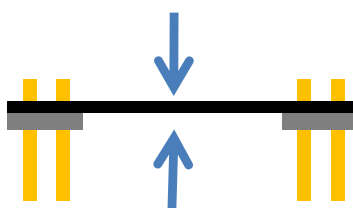
ハンダ付けの位置

裏面 (*35) からCN1,2に、2列ピンヘッダの**短い方**を差し込み
表側からハンダ付けする



裏面

基板の表



基板の裏

(*33)
作業机はいつもキレイに!



ATTENTION!!

I/Oボードやマイコンボードを置くときには、ボードの下にカットした電子部品の足やハンダのカスなどがなくよく確認し、置く場所をきれいにしましょう。

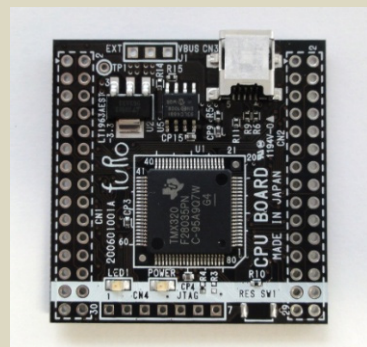
裏側の端子と金属片などが接触すると、回路がショートしてマイコンやI/Oボードが壊れる可能性があります。

(*34)
ピンヘッダの準備

2列ピンヘッダは縦2列×横15列=30ピンが2つできるように、予めカットしておきましょう。カットの仕方はP8「ピンヘッダのカット」の項を参照のこと。

(*35)
マイコンボードの表裏の見分け方

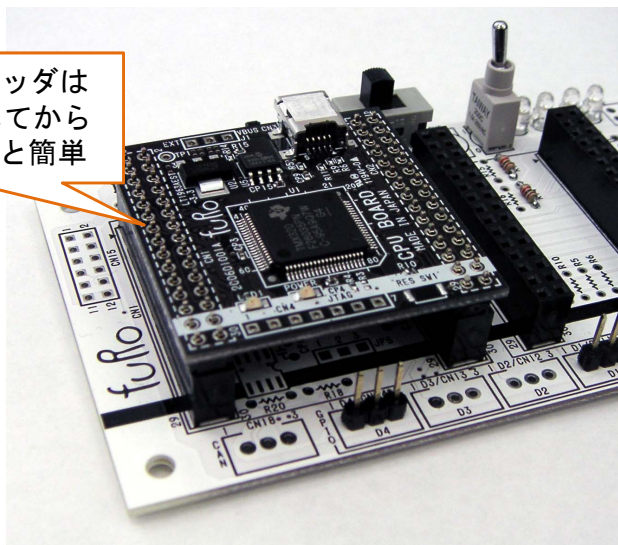
マイコンボードの表には、
①「fuRo」の文字が書いてある
②マイコンが搭載されている
③USBコネクタが付いている



こっちが表です。

今回はピンヘッダを差し込むためのコネクタがすでにI/Oボードにハンダ付けされているので、そこにピンヘッダを差し込み、その上にマイコンボードを差し込んで固定してからハンダ付けすると、曲がらずに簡単にハンダ付けすることができます。

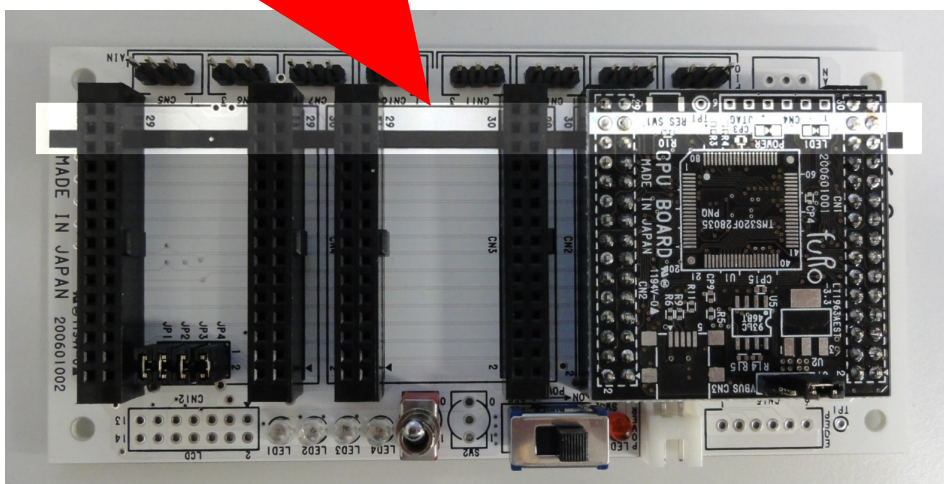
30ピンのピンヘッダは
I/Oボードに刺してから
ハンダ付けすると簡単



I/Oボードへのとの接続

マイコンボードの差す向きに注意して、I/Oボードに差し込んでください。(*36)

I/Oボードにプリントされている黒い線と、
マイコンボードにプリントされている白い線が
一直線になるように合わせて差し込もう。
逆にすると、マイコンが燃えるよ！

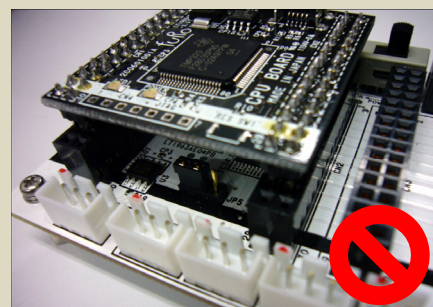


⚠ **ATTENTION!!** ⚠

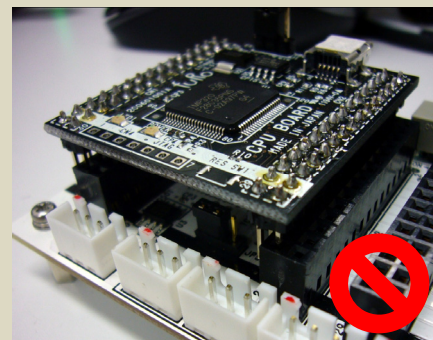
(*36)
ボードの差し込みは慎重に

⚠ **ATTENTION!!**

マイコンボードをI/Oボードに差し込む時には、黒い線と白い線を揃えるのはもちろん、マイコンボードのピンヘッダとI/Oボードのコネクタがずれないように気をつけましょう。ずれて差し込んだまま電源を入れると**マイコンが壊れます**。



駄目な例1: 縦方向にずれている



駄目な例2: 横方向にずれている

4章 開発環境

概要

ここでは開発環境について学びます(*45)。2章で組み立てたf-paletteにプログラムを書き込み、実際に動かすための環境と考えてください。まずは、ソフトウェアのインストールと使い方です。

- 4-1 開発環境「CCS」のインストール
- 4-2 サポートファイルのインストール
- 4-3 開発環境「CCS」の使い方
- 4-4 Arduino互換の開発環境「f-palette IDE」のインストール

4-1 開発環境「CCS」のインストール

① 開発環境「CCS」について

ロボットを思い通りに動かすため、マイコンに「プログラム」を書き込みます。プログラムを書き込むソフトはマイコンによって違います。

f-paletteはTI(テキサス・インスツルメンツ)社のマイコン(Piccolo)を使用しているため、TI社の「CCS v4(Code Composer Studio version 4)」というソフトウェアを使います。(*46)

このソフトの中にはコンパイル(翻訳)、デバッグ(間違い探し)など多数の機能が含まれており、これら全てをまとめて開発環境とよびます。(コンパイル、デバッグについては後で説明します)

CCS v4は無料で入手できます。TI社のウェブサイト

http://processors.wiki.ti.com/index.php/Download_CCS

の「Contents」リスト中、

2.2 Download latest production CCSv4 MSP430/C28x code size limited image をクリックすると、下記の画面に移ります。この [ダウンロード] ボタンをクリックします。

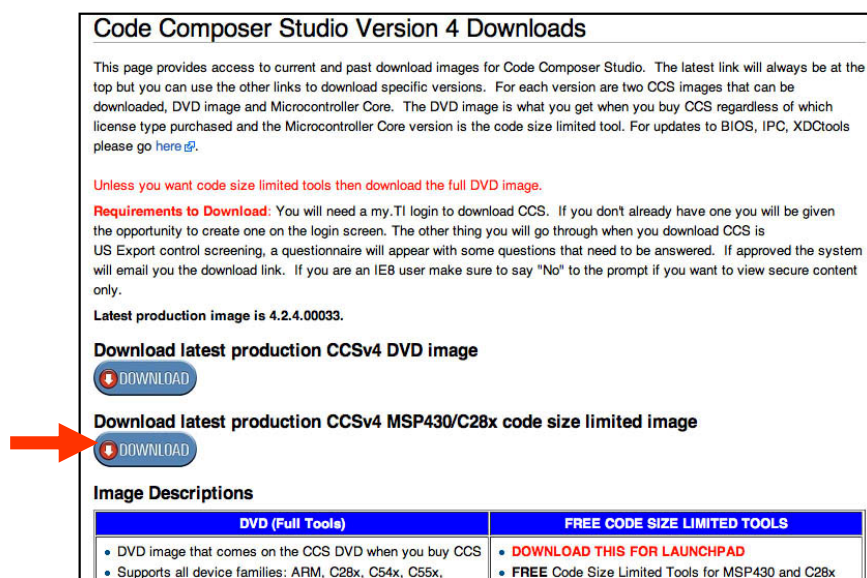


図4-1-1 TI社ウェブサイトの画面

(*45)

2つの開発環境について

f-paletteを使ってプログラム開発をする方法は2つあります。プログラム開発の習熟度や慣れによって選択するのがいいでしょう。

本格的にロボットの開発をしたい人や、C言語でバリバリ開発したい人は「CCS」を使いましょう。もっと簡単に開発したい人、Arduino環境に慣れている人には「f-palette IDE」をお勧めします。

なお、f-palette IDE環境で開発したい人も、CCSをインストールする必要があります。

(*46)

TI社によるCCSの解説

Code Composer Studio はTI社のエンベデッド・プロセッサ向けの統合開発環境です。Code Composer Studio には、組み込みアプリケーションの開発とデバッグに必要なツールが含まれています。TIの各デバイス・ファミリ向けのコンパイラ、ソース・コード・エディタ、プロジェクト・ビルド環境、デバッグ、プロファイラ、シミュレータなど、多数の機能が含まれています。

バージョン 4 以降の Code Composer Studio は、Eclipse オープン・ソース・ソフトウェア・フレームワークに基づいています。Eclipse は、多数の組み込みソフトウェア・ベンダによって使用される標準フレームワークになってきています。

以上、TI社ウェブサイト

http://www.tij.co.jp/lsds/ti_ja/dsp/support/dev_tool/ccs_overview.page
より抜粋。

② ユーザ登録とログイン

ダウンロードの前に、まずユーザ登録を行う必要があります。[ダウンロード] ボタンをクリックすると、自動的に登録画面に飛びます。

図4-1-2 TI社のユーザ登録画面

ここで入力するのは、以下の情報です。

- * Your country/region (「JAPAN」を選択するとテキストが日本語にかわります)
- * eメール・アドレス (メールアドレスを入力)
- * パスワード (次回以降ログインするために必要なパスワードを入れます。6文字から12文字です。)
- * パスワードの確認入力 (上記パスワードの確認です。同じ文字を入れます)
- * 姓 (苗字を入力)
- * 名 (名前を入力)
- * 会社名 (所属を入力、会社名か大学名)

ローマ字で入力

- * Last Name (苗字を入力)
- * First Name (名前を入力)
- * Company (所属を入力、会社名か大学名)

「登録情報を記録する」「最新情報をeメールで受信する」にチェックを入れましょう。
最後に「登録」を押します。

登録したメールアドレスに「my.TI 登録 Eメールアドレス認証手続き」というメールが届きますので、リンクをクリックしてEメールアドレスを認証してください。

「ご登録ありがとうございました。電子メール・アドレスの確認が完了しました。」と画面に表示されたら、[TI.com ホームに移動]または[my.TI アカウント・ホームに移動]のどちらかをクリックします。開いたページの右上、「TIログイン」にメールアドレスとパスワードを入力して、ログインします。

③ ダウンロード

先ほどのダウンロードページ

http://processors.wiki.ti.com/index.php/Download_CCS

を再び開いて

2.2 Download latest production CCSv4 MSP430/C28x code size limited image
の[ダウンロード]をクリックします。
すると次のような入力画面が現れます。

TEXAS INSTRUMENTS

Products Applications Design Support Sample & Buy All Searches Search by part number or keyword GO

TI Home

TI Software

To download:

- If you are approved, a DOWNLOAD BUTTON will appear.
- If you are not immediately approved, a message will appear after this form; Software may be delayed 1-2+ business days.
- This software requires U.S. Government export approval before download.
- To AVOID delays, please provide complete information and do NOT use abbreviations
- We apologize for any inconvenience.

U.S. Government export approval: All fields are Required

First name: [text box]

Last name: [text box]

Your email address: [text box]

Confirm email address: [text box]

Your full company/university name: [text box]

Your company/university website: [text box]

Country this software will be used in: [dropdown menu]

What end-equipment/application will you use this software for:

<input type="checkbox"/> AV Receivers	<input type="checkbox"/> Biometrics
<input type="checkbox"/> Digital Still Camera/DVR	<input type="checkbox"/> Military
<input type="checkbox"/> Portable Media Devices	<input type="checkbox"/> Radar
<input type="checkbox"/> Security	<input type="checkbox"/> Streaming Media
<input type="checkbox"/> Video Conferencing	<input type="checkbox"/> Video Infrastructure Broadcast Video
<input type="checkbox"/> VoIP Solutions	<input type="checkbox"/> WiMAX/Wireless

Other: [text box]

IMPORTANT: Provide a brief abstract of how this software will be used: [text box]

(Your request may be denied if this information is incomplete.)

図4-1-3 ダウンロード用入力画面

ここでは以下の情報を入力します。

First Name (名前を入力)
Last Name (苗字を入力)
Your email address (メールアドレスを入力)
Confirm email address (確認のため、再度同じメールアドレスを入力)
Your full company/university name (会社名または大学名を入力)
Your company/university website: (上記のURLを入力)
Country this software will be used in (国を選択)
What end-equipment/application will you use this software for (ソフトの使用目的を選択)
IMPORTANT: Provide a brief abstract of how this software will be used (最終的に何に使うかを記入)
I certify that the following is true:

I CERTIFY ALL THE ABOVE IS TRUE:

内容を読んで、すべて真実であれば、Yesを選択して、[Submit]をクリックするとダウンロードに進みます。

セキュリティの警告はすべて[いいえ]をクリックしてください。

[いいえ]をクリックし、ブラウザの上の方に「セキュリティ保護のため——」というテキストがでてきたら、右クリックで「ファイルのダウンロード」を選択します。保存先を指定して、ファイルをダウンロードしてください。(*47)

(*47)

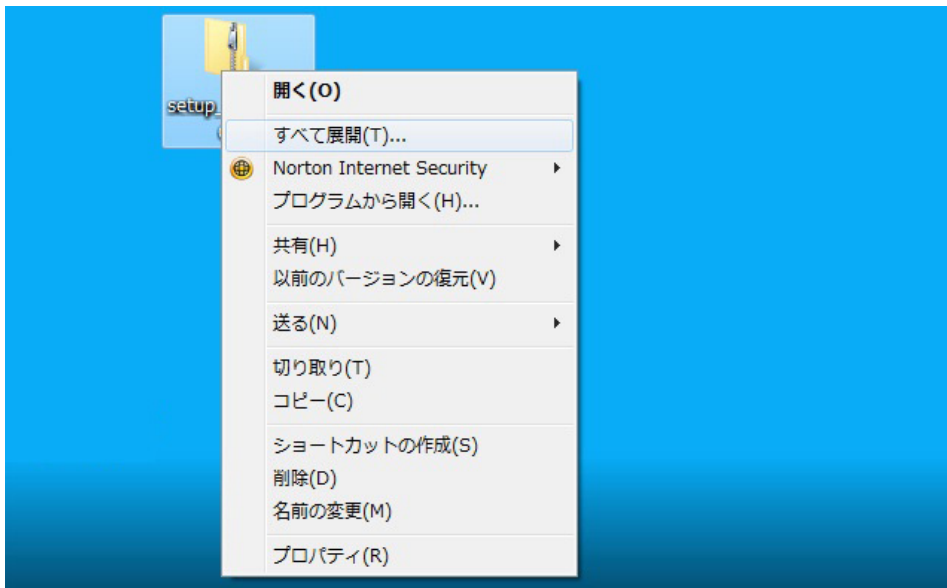
別のダウンロード方法

「TI SOFTWARE DOWNLOAD:
APPROVED - setup_CCS_MC_Core.zip
」

というメールが届いていたら、こちらからもファイルのダウンロードができます。左記のようにセキュリティの警告はすべて「いいえ」をクリックしてください。「いいえ」をクリックし、ブラウザの上の方に「セキュリティ保護のため——」というテキストがでてきたら、右クリックをして「ファイルのダウンロード」を選択、保存先を指定して、ファイルをダウンロードしてください

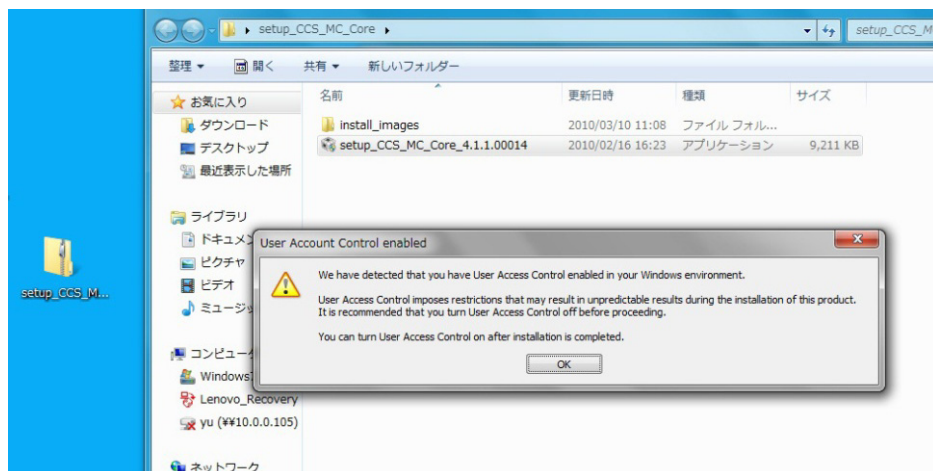
④ インストール

ダウンロードしたファイルは、「setup_CCS_MC_Core.Zip」という圧縮ファイルです。これをデスクトップにコピーします。コピーが完了したら、右クリックで「すべて展開」を選択し、ファイルを解凍します。

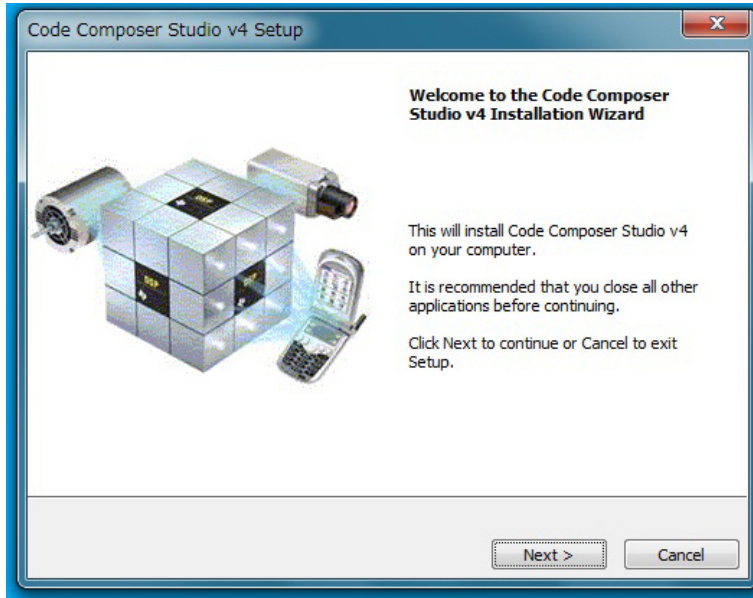


解凍すると「setup_CCS_MC_Core」というフォルダが現れます。この中に「setup_CCS_MC_Core_xxxx.exe」というファイルがあります。(xxxxはバージョン番号。最新版をインストールしましょう)

これをダブルクリックすると、自動的にインストールが始まります。

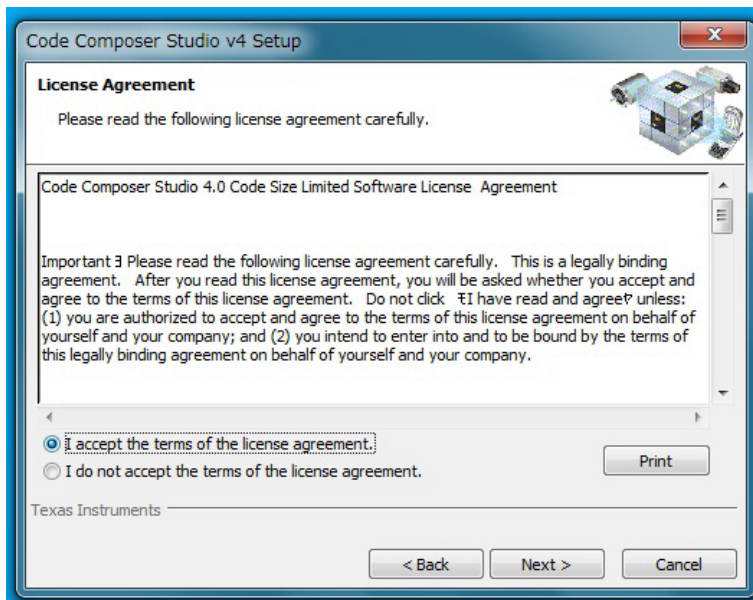


[OK]をクリックしてください。以下、インストールの流れに沿って手順を解説します。



【Code Composer Studio v4 Setup】ウィンドウ
Welcome to the Code Composer
Studio v4 Installation Wizard

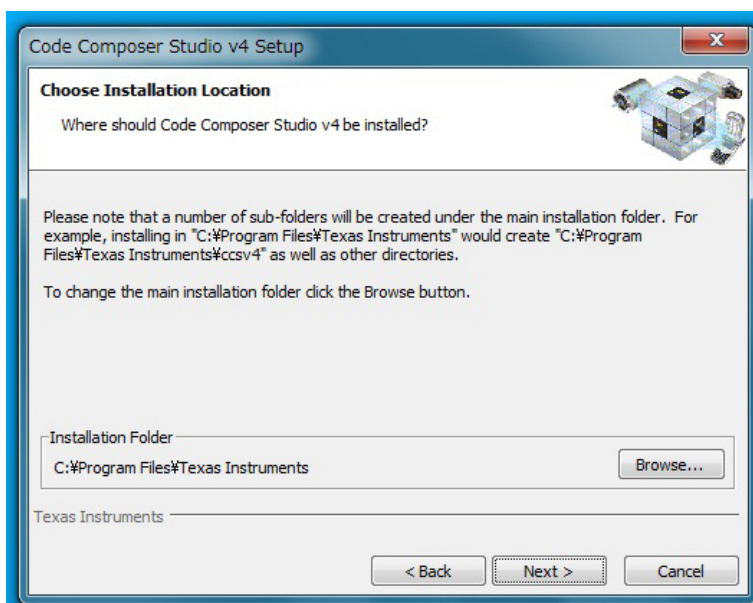
セットアップ・ダイアログボックスが表示されますので
[Next >]をクリックしてください。



【Code Composer Studio v4 Setup】ウィンドウ
License Agreement

ソフトウェア・ライセンス契約が表示されます。表示を
スクロールさせて文面を確認し、契約に同意される
場合は、「I accept the terms of the license
agreement」を選択し、[Next >]ボタンをクリックし
てください。

(契約に同意できない場合は、[Cancel] ボタンをク
リックしてセットアップを中止してください。)



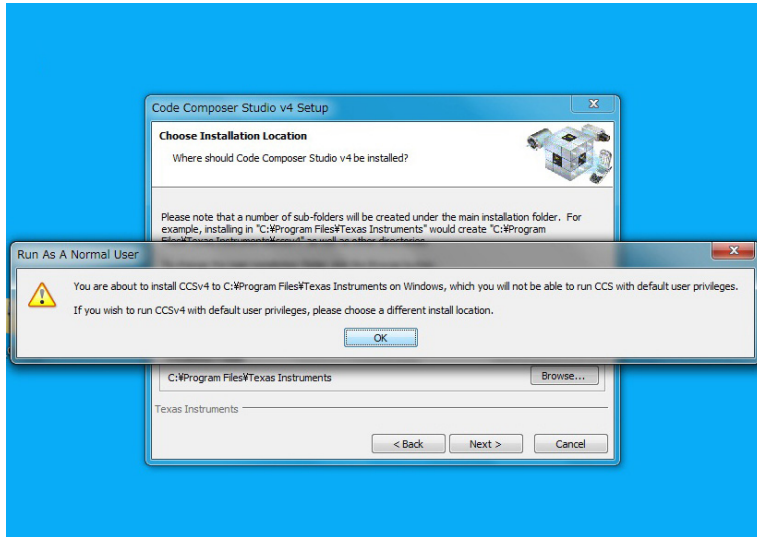
【Code Composer Studio v4 Setup】ウィンドウ
Choose Installation Location

CCSv4 のインストール先を指定する画面が表示され
ます。デフォルトでは下記のいずれかになります。

C:\Program Files\Texas Instruments

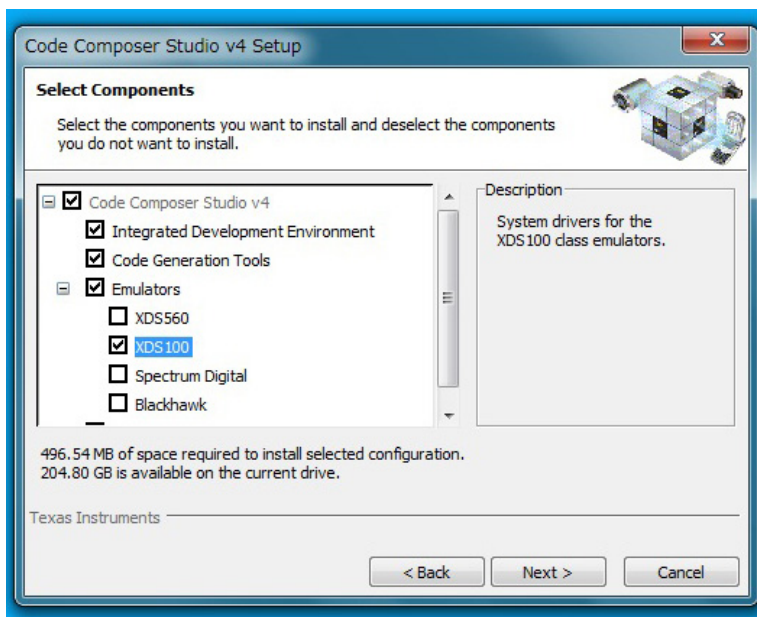
C:\Program Files (x86)\Texas Instruments

フォルダを決定後、[Next >]ボタンをクリックして
ください。以降の説明はデフォルトの設定で行います。



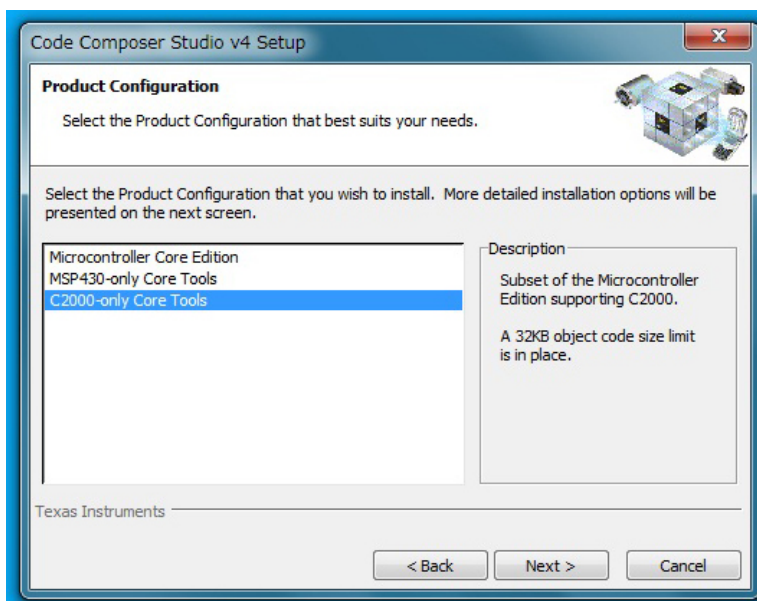
【Run As A Normal User】ウィンドウ

[OK]をクリック



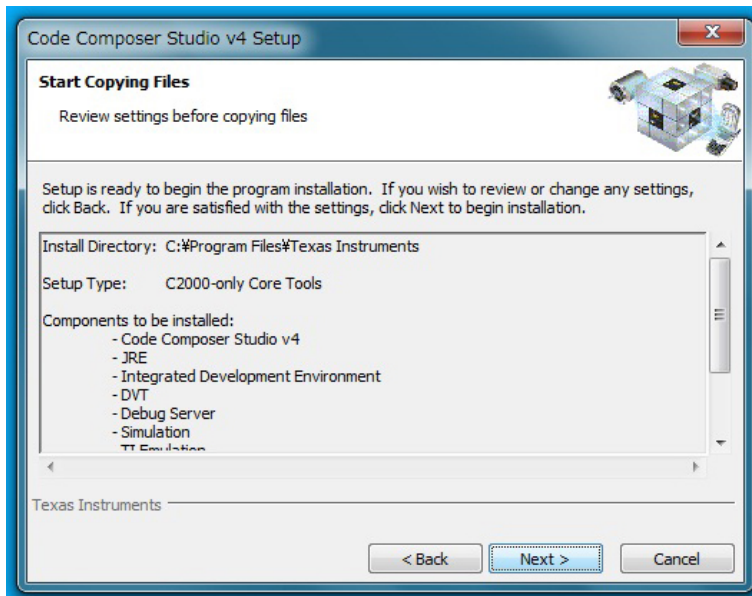
【Code Composer Studio v4 Setup】ウィンドウ
Select Components

Emulatorsの中の「XDS100」以外のチェックをはずして、
[Next >]ボタンをクリックしてください。



【Code Composer Studio v4 Setup】ウィンドウ
Product Configuration

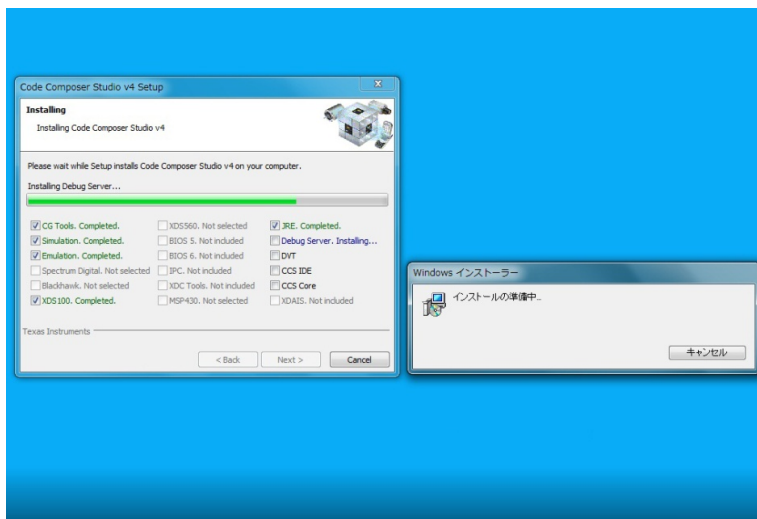
「C2000-only Core Tools」を選択



【Code Composer Studio v4 Setup】ウィンドウ Start Copying Files

インストールの準備が完了すると、確認画面が表示されます。

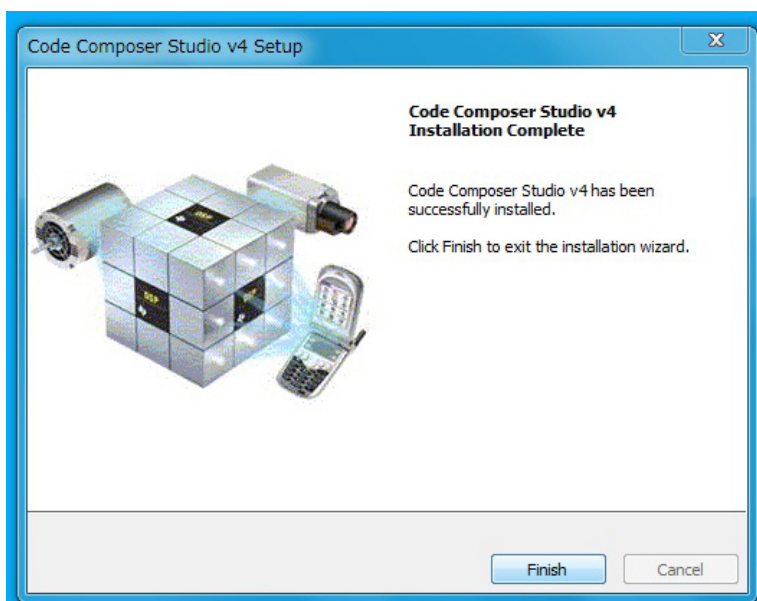
インストール先フォルダ名、製品名、必要なコンポーネントを確認して [Next >] ボタンをクリックしてください。



【Code Composer Studio v4 Setup】ウィンドウ Installing

ファイルのコピーが始まり、インストール中のコンポーネントが表示されます。

コピーが完了するまで、しばらくお待ちください。インストールに要する時間は、PC の性能に依存します。



【Code Composer Studio v4 Setup】ウィンドウ Code Composer Studio v4 Installation Complete

CCS v4 のインストールが正常に完了した場合は左の画面が表示されます。

[Finish] ボタンをクリックしてインストールを終了してください。デスクトップ上に、下記のようなCCS v4のサムネイルができています。



4-2 サポートファイルのインストール

次に、開発時に有用なサンプルプログラムなどのサポートファイルをインストールします。

① ダウンロード

サンプルファイルはTI社の次のサイトからダウンロードできます。

<http://www.tij.co.jp/tool/jp/sprc892>

ページ中の型番「SPRC892: 2803x C/C++ ヘッダー・ファイルとペリフェラルの例」にある[ダウンロード]をクリックすると、自動的に「sprc892.zip」という圧縮ファイルがダウンロードされます。

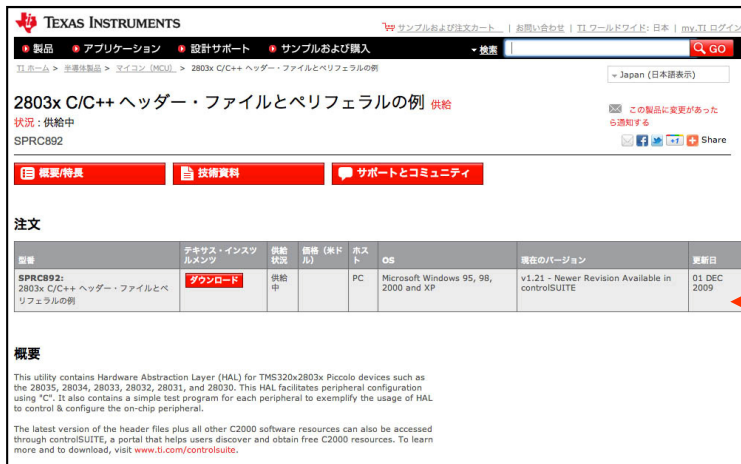
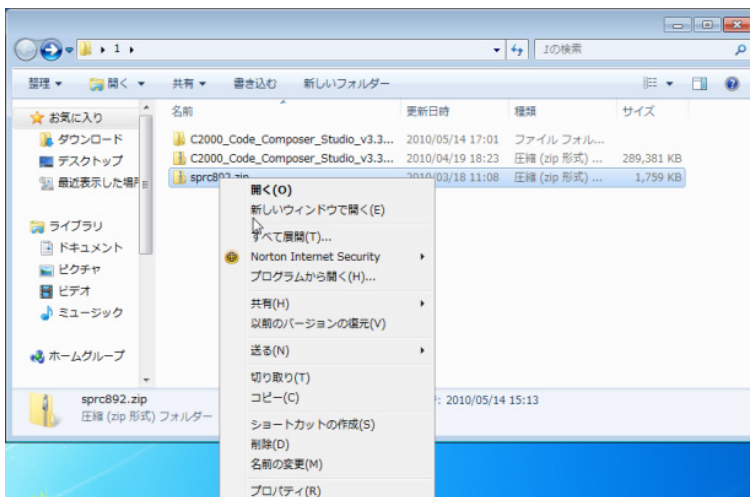


図4-2-1 ダウンロードページ

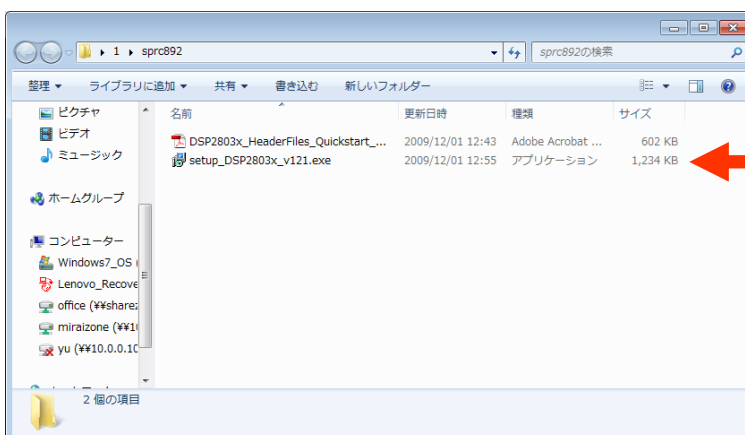
ダウンロードが完了したら、右クリックで「すべて展開」を選択し、任意の展開先を指定して、ファイルを解凍します。

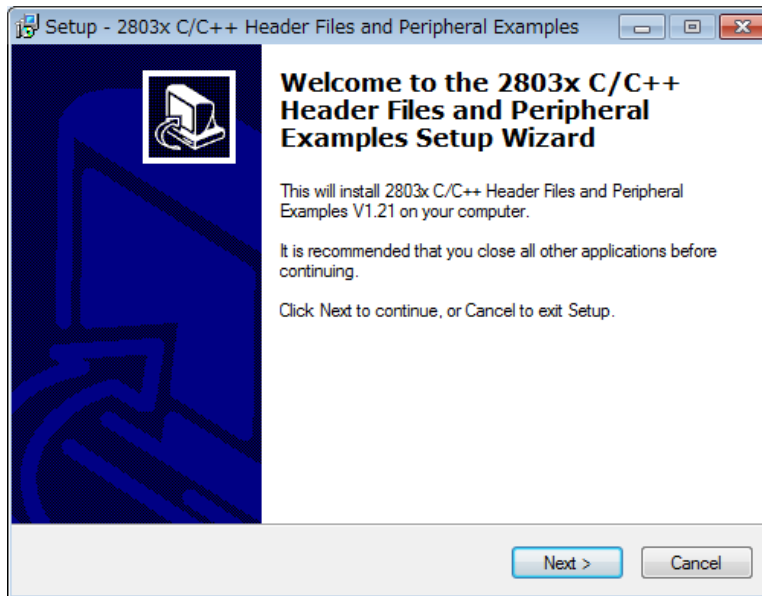


② インストール

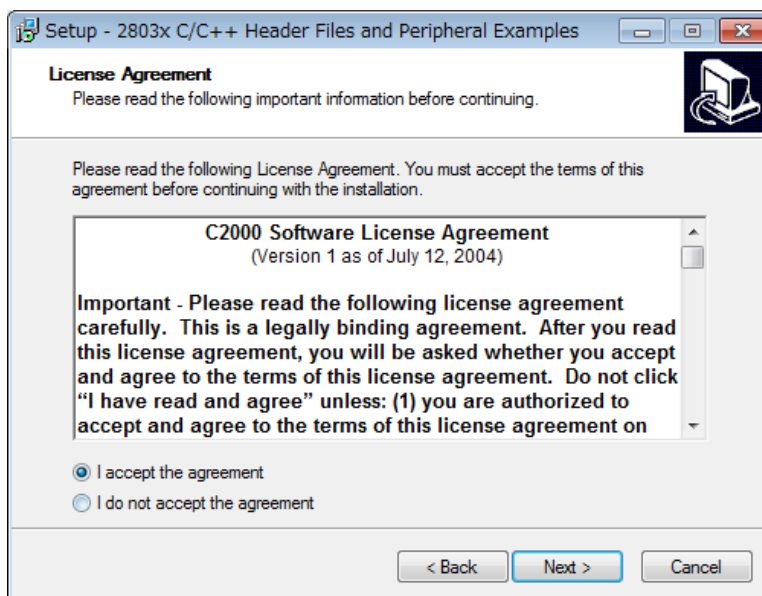
展開したフォルダ「sprc892」の中に、「setup_DSP2803x_v121.exe」というファイルがあります。

これをダブルクリックするとインストールが始まります。

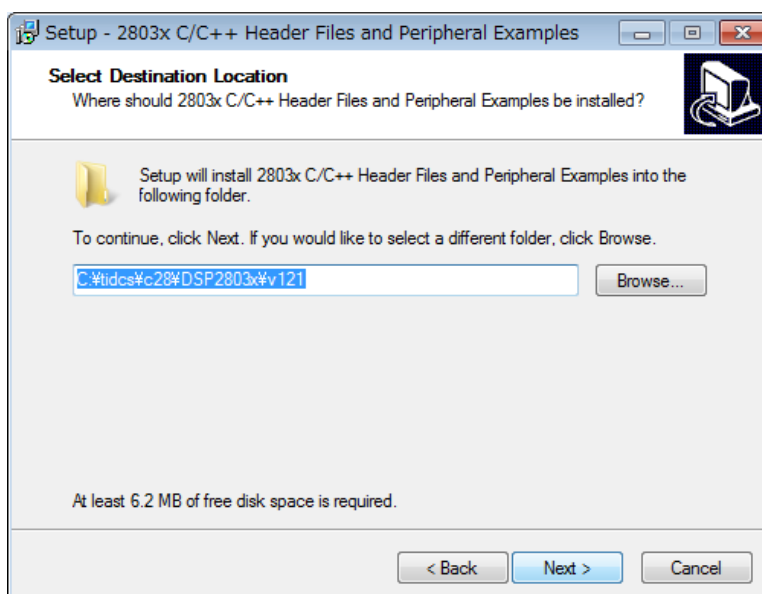




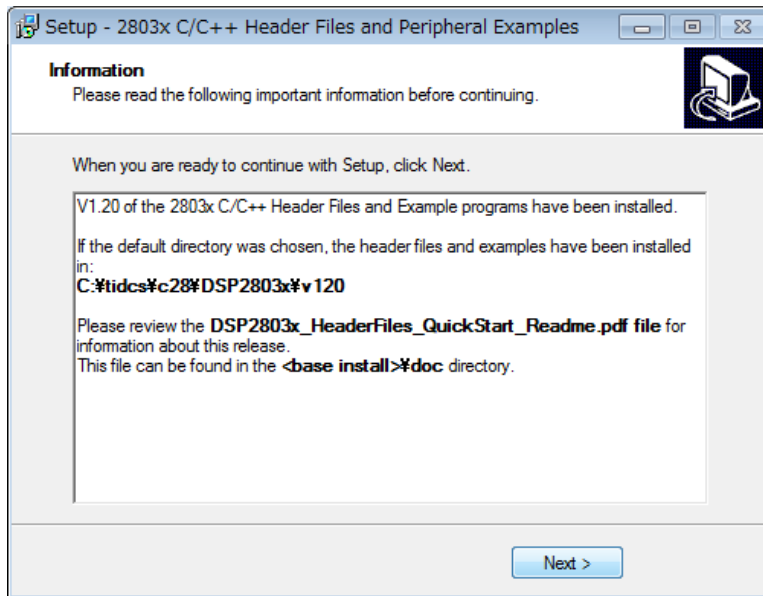
[Next>]をクリックし、



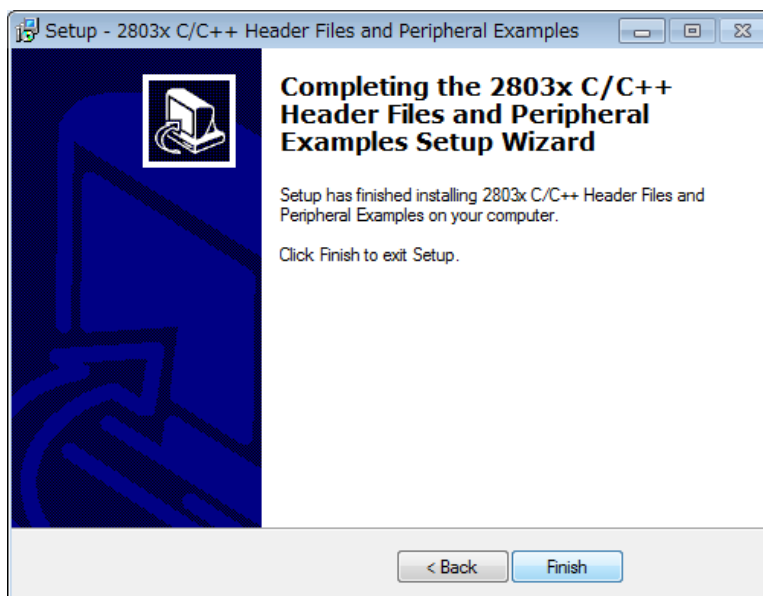
[I accept the agreement]にチェックを入れて、[Next>]をクリック。



[Next>]をクリック



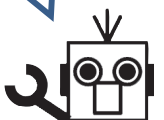
[Next>]をクリック



[Finish]をクリック

これで開発環境の準備は完了です。いよいよ、マイコンのプログラミングに移ります。

これで準備は終わりダ！
いよいよ、ボクの脳ミソを作る
時が来タ！



4-3 開発環境「CCS」の使い方

インストールした開発環境を使って、組み立てたf-paletteを実際に動かしてみましょう。開発環境にサンプルプログラムを読み込み、プログラムを書き換え、プログラムをコンパイルして、マイコンとケーブルで接続し、プログラムを書き込み、動作確認するまでの一連の流れを習得します。

- ① マイコンプログラム開発手順
- ② CCS v4の準備
- ③ マイコンの選択
- ④ プログラムの読み込み
- ⑤ プログラムの書き換え
- ⑥ マシン語に翻訳、ファイルをつくる
- ⑦ マイコンとコンピュータをつなぐ
- ⑧ プログラムの間違い探し
- ⑨ プログラムを送る
- ⑩ 動かす
- ⑪ とりはずす

① マイコンプログラム開発手順

開発環境(CCS)を使ってパソコン上でプログラムを組み立て(Build)、ケーブルを介してマイコンに読み込ませ(Load)、実行(Run)します。

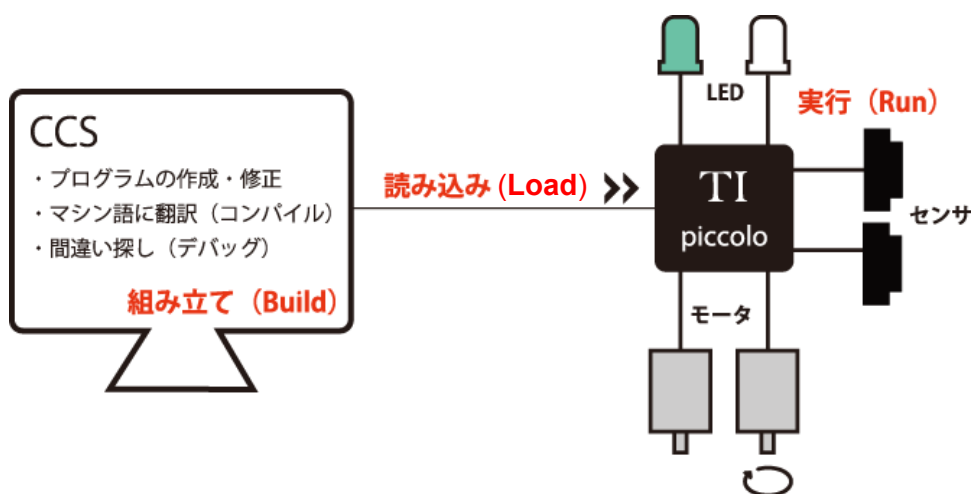


図4-3-1 マイコンプログラム開発手順

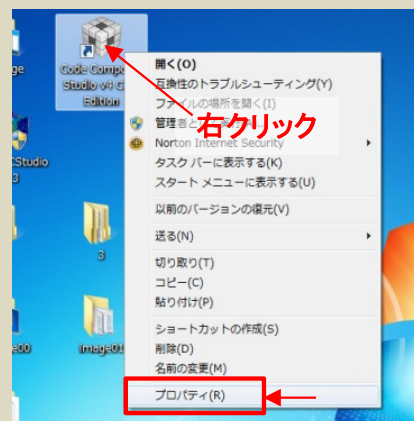
② CCS v4の準備

最初の1回目だけ、CCSの管理者権限実行設定を行う必要があります。デスクトップ上のCSS v4アイコンのプロパティを変更します(*48)。

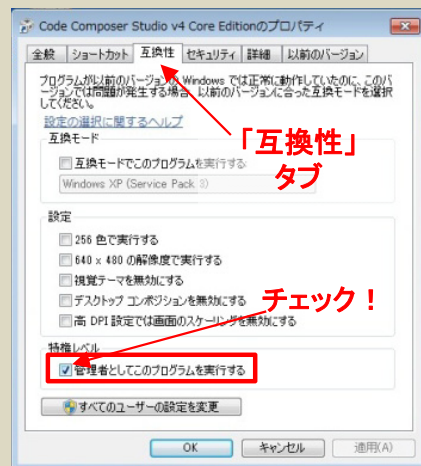
(*48)

CCSの管理者権限実行設定

デスクトップの「Code Composer Studio v4 Core Edition」を右クリックして、プロパティを選択。

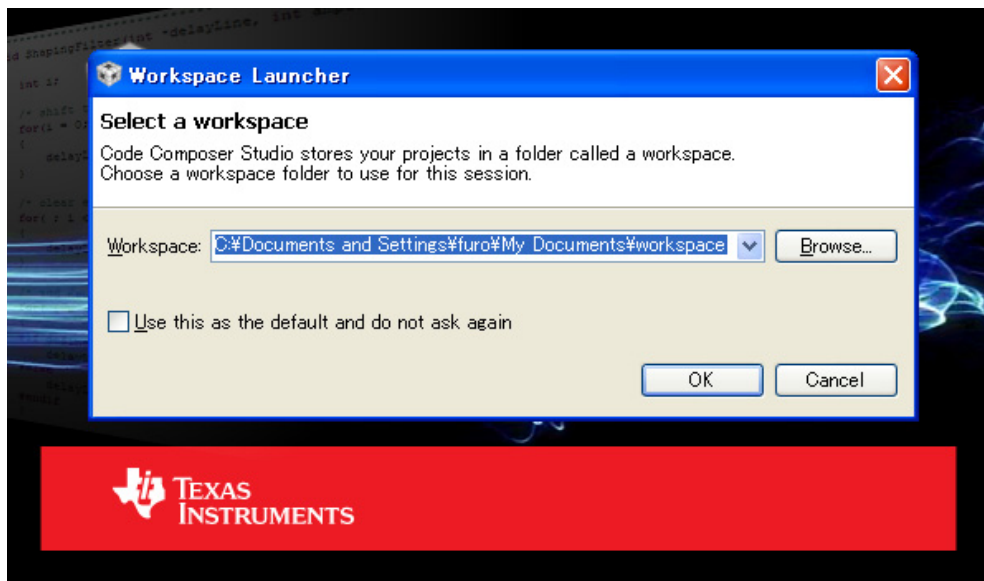


「互換性」タブを選択し、下にある「特権レベル」の「管理者としてこのプログラムを実行する」にチェックをいれる。



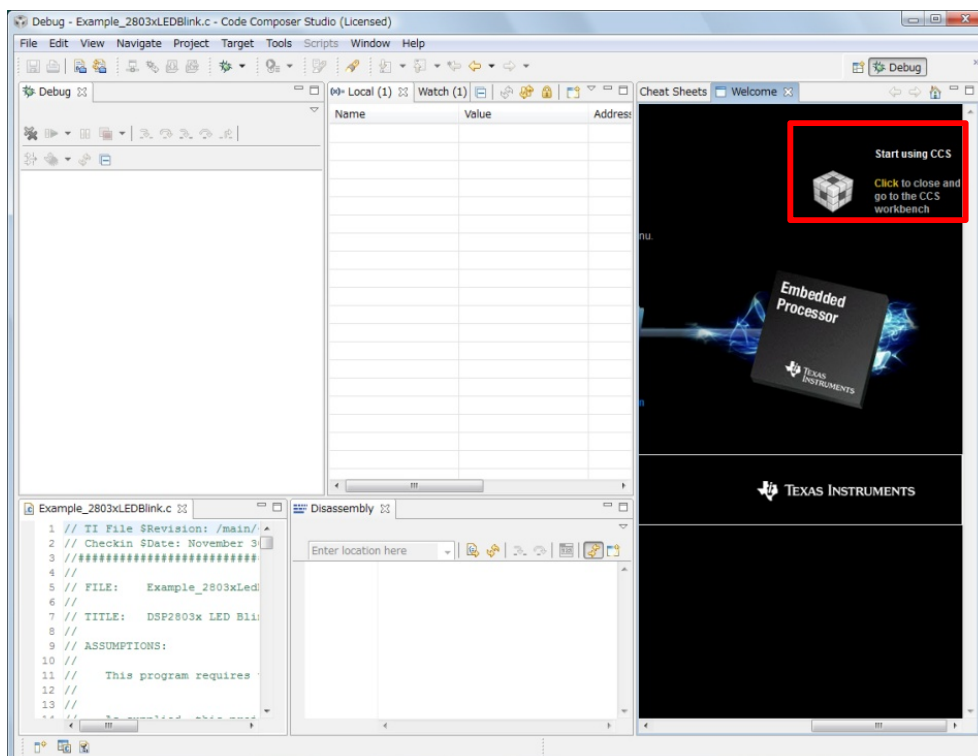
CCS v4のアイコンをダブルクリックして起動します。最初にこの開発環境を動かすための「ワークスペース」を設定するウィンドウが現れます。

【Workspace Launcher】ウィンドウ



ワークスペースというのは作業机みたいなものです。作業に必要な書類や部品やらを広げる場所ですね。場所に問題がなければそのまま[OK]をクリックします。(*49)

ワークスペースは、作業途中でも[File→Switch WorkSpace]で変更できます。または [Start using CCS]をクリックしても変更できます。



(*49) ワークスペースについて

Windows7の場合、ワークスペースは、
C:\Users\furo\Documents\workspace
に自動設定されます。

なお、ワークスペースパスの途中に日本語が入っていると正常動作しません。
たとえば指定が、

C:\Documents and Settings¥太郎¥My Documents¥workspace

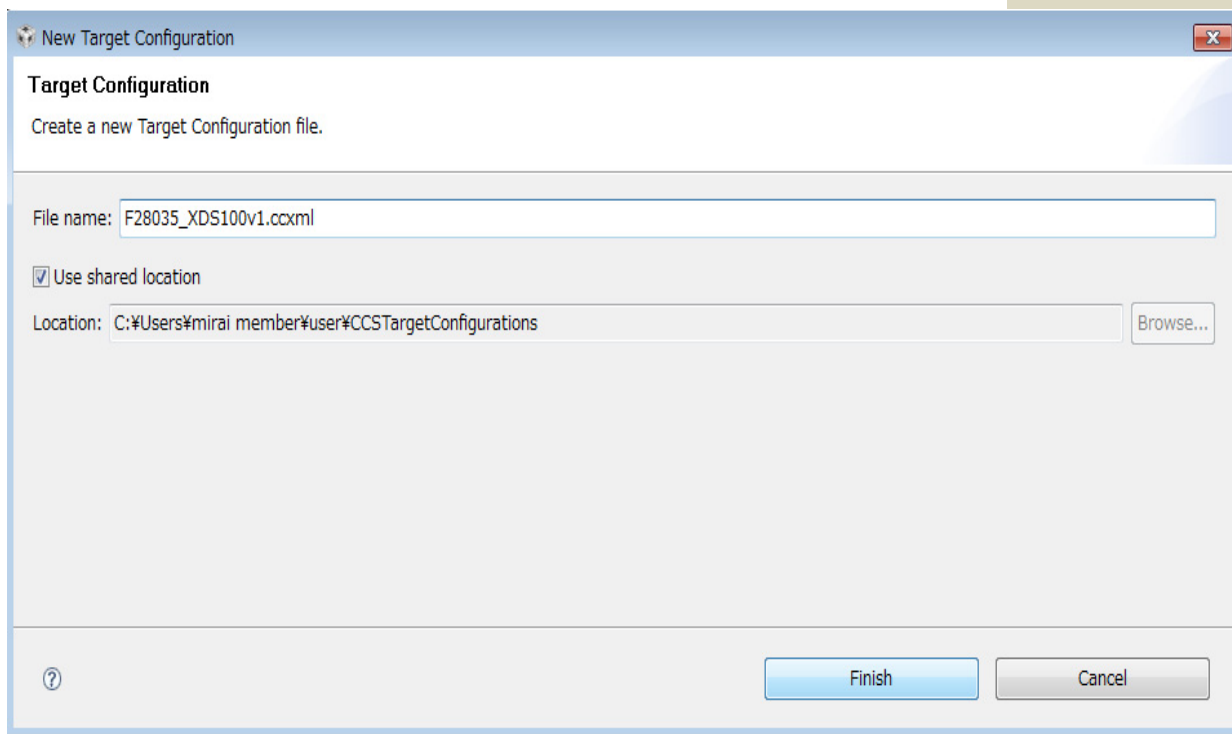
の場合うまく動かないことがあります。

③ マイコンの選択

これからプログラムを書き込むマイコンを設定します。これは最初の一度だけの作業で、2回目以降は必要はありません。

[Target]→[New Target Configuration](※50)

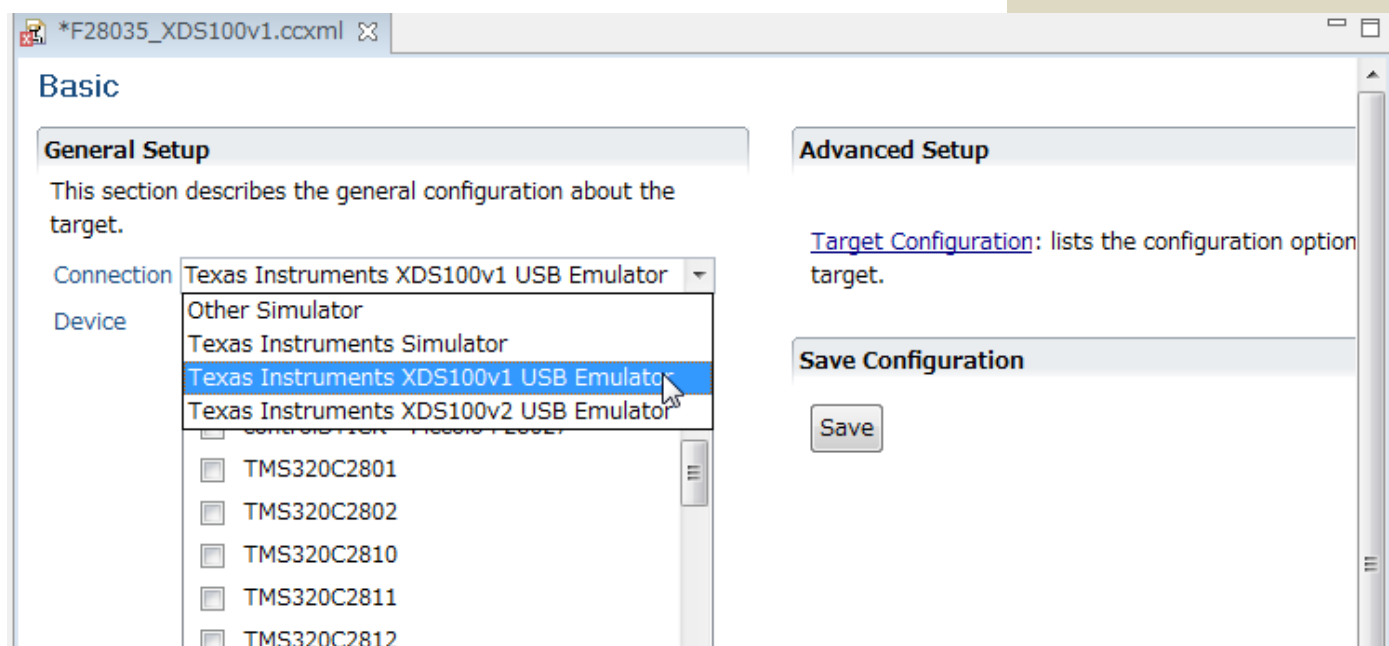
【New Target Configuration】ウィンドウ



File nameを「F28035_XDS100v1.ccxml」としてください。

[Finish]をクリック

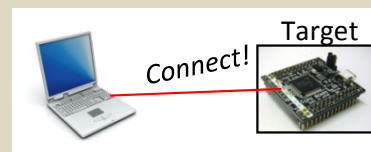
【*F28035_XDS100v1.ccxml】タブ

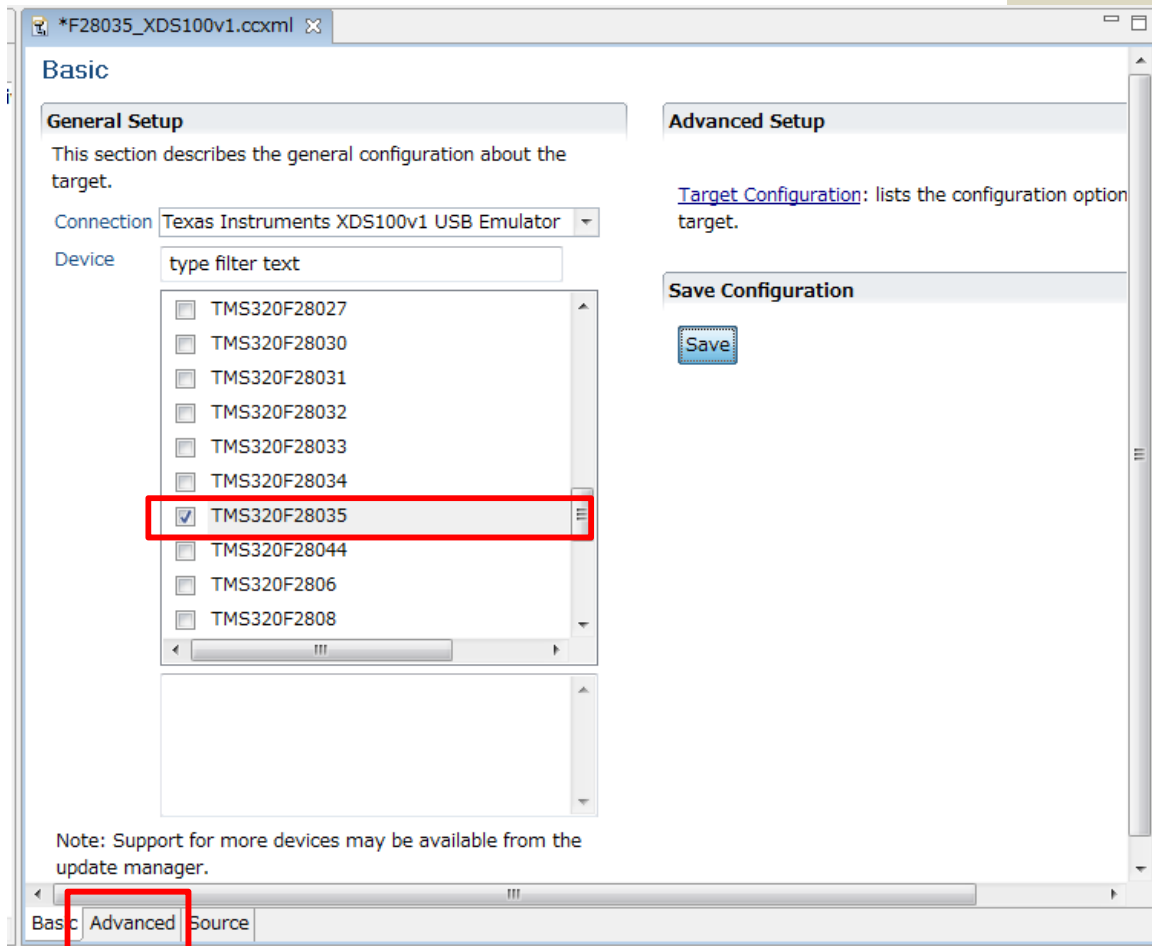


Connection [Texas Instruments XDS100v1 USB Emulator]を選択

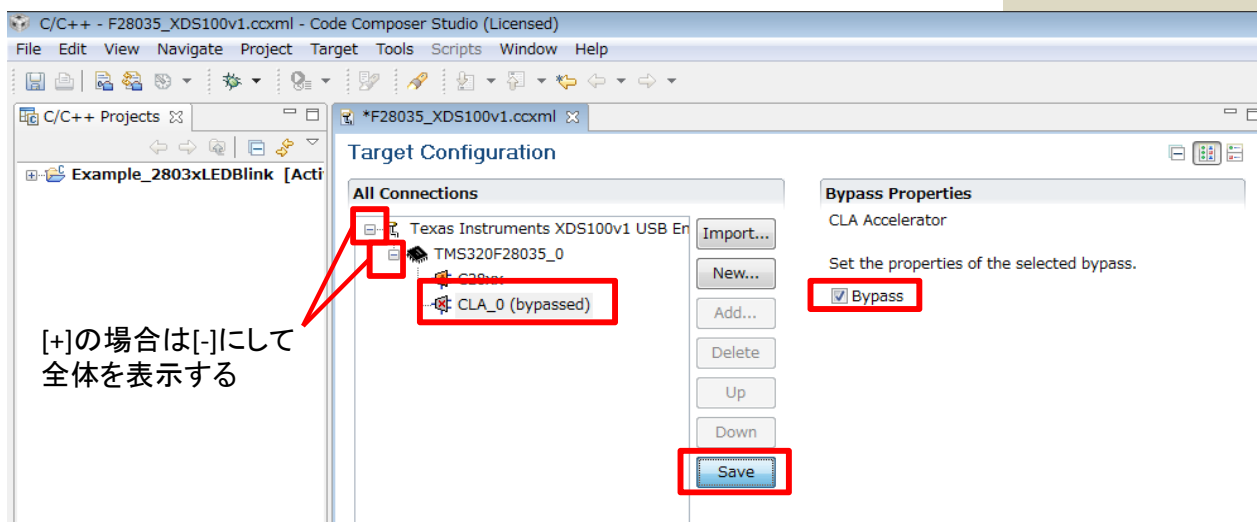
(※50)
開発環境からみたターゲット

ここでいう「Target」とはマイコンのことです。





Device[TMS320F28035]を選択し(*51)、下段タブ [Advanced]を選択



[CLA_0(bypassed)]をクリック、右ウィンドウ、[Bypass]をチェック
[Save]をクリック

これで、プログラムを書き込むマイコンを選定しました。マイコンによっては、命令の伝え方もしやり方も変わってきます。今の作業で、

「この方はどこから来た～さんだから～で翻訳して伝えてよ」
という伝票を作ったと思ってください。

(*51)
マイコンの機種を選択

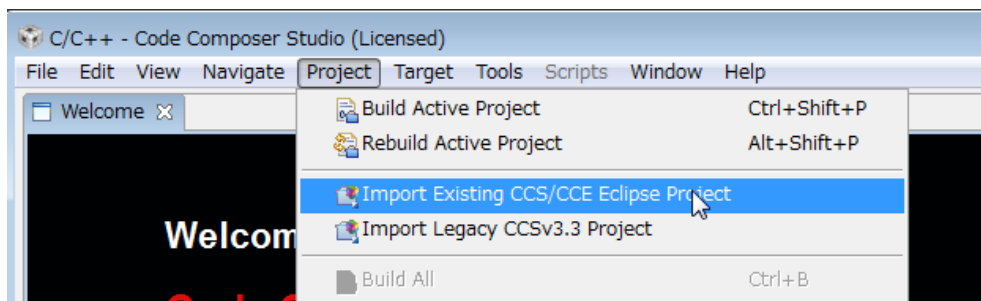
f-paletteにはこの「TMS320F28035」という名前のマイコンが使われています。

④ プログラムの読み込み

まず、「プロジェクト」のインポートを行います。

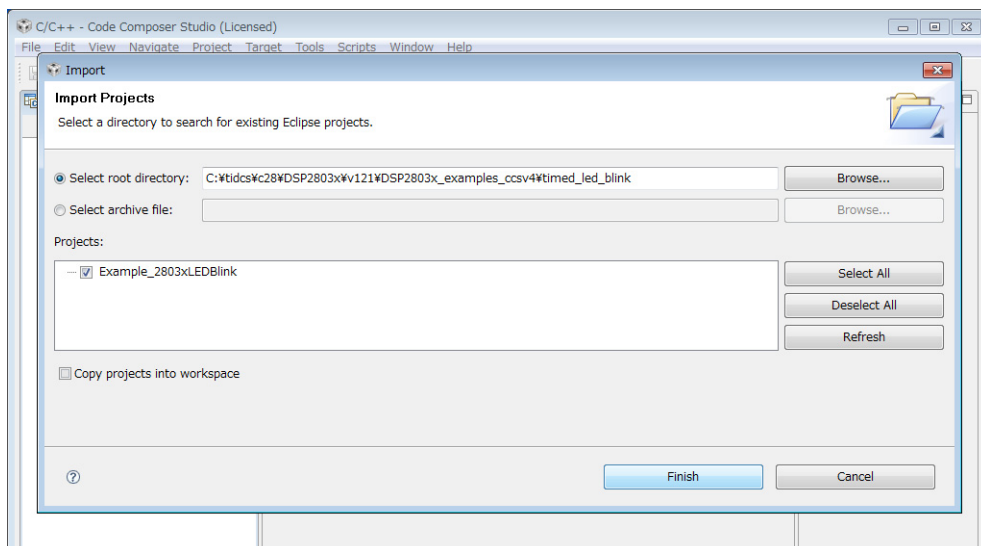
いくつかのプログラムのかたまりを「プロジェクト」といいます。プロジェクトのインポートとは、机の上に本棚からファイラーを一式持ってくるようなことです。
(*52)

【C/C++- Code Composer Studio(Licensed)】ウィンドウ



「Project」→「Import Existing CSS/CCE Eclipse Project」を選択。(*53)

【Import】ウィンドウ



[Select root directory]をチェックし、[Browse]をクリックして呼び出したいプロジェクトを選択します。今回はLED1を点灯させるサンプルプログラム、

[C:\tidcs¥c28¥DSP2803x¥v121¥DSP2803x_examples_ccsv4¥timed_led_blink]

を選択します。[Copy projects into workspace]は今回はチェックせずに、[Finish]をクリック。

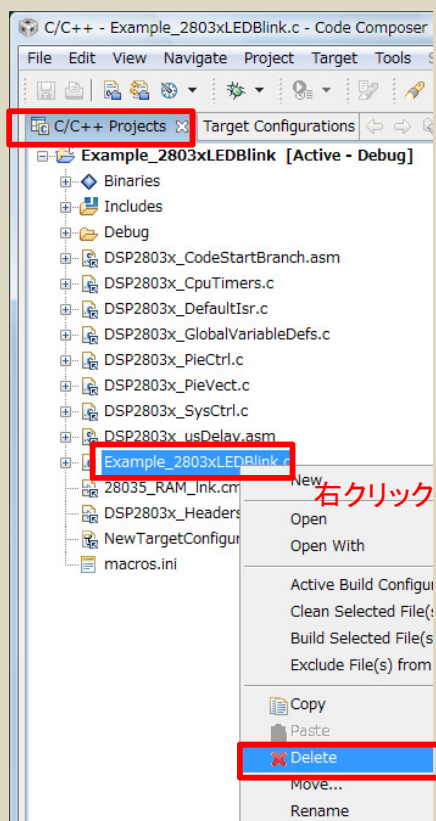
これで本棚から翻訳すべき本(プロジェクト)を持ってきたことになります。

次は、これは「どこから来た～さんと～語を翻訳します。」という伝票を本に貼り付けます。すなわち、③で指定したマイコンと、いま呼び出したプロジェクトを結びつけるということです。

(*52)<注意>

ファイルの閉じ方

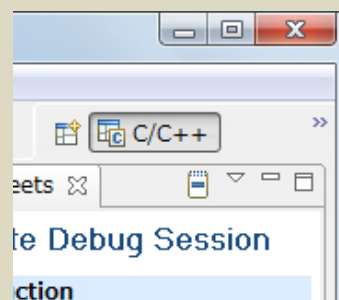
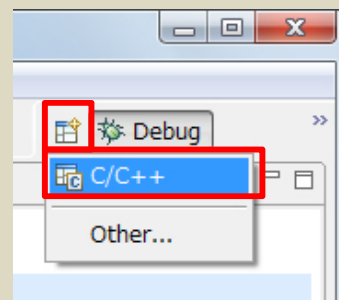
同じ名前のファイルをすでに開いている場合は開けません。一度閉じて再度開いてください。ファイルの閉じ方は、[C/C++Projects]タブを選択し、消したいファイルを右クリックし[Delete]を選択します。

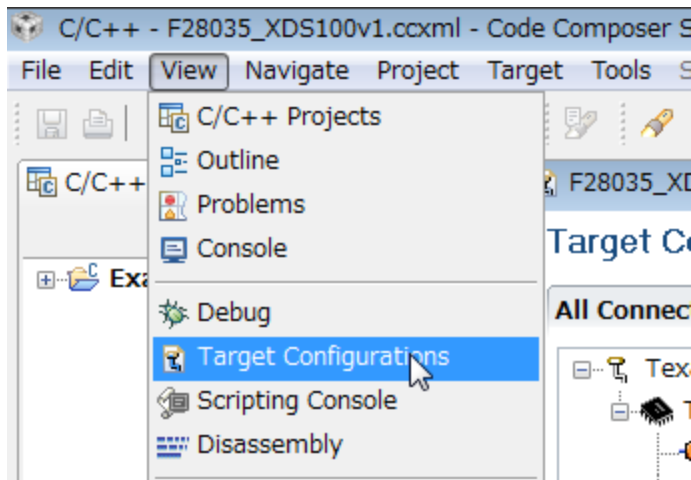


(*53)

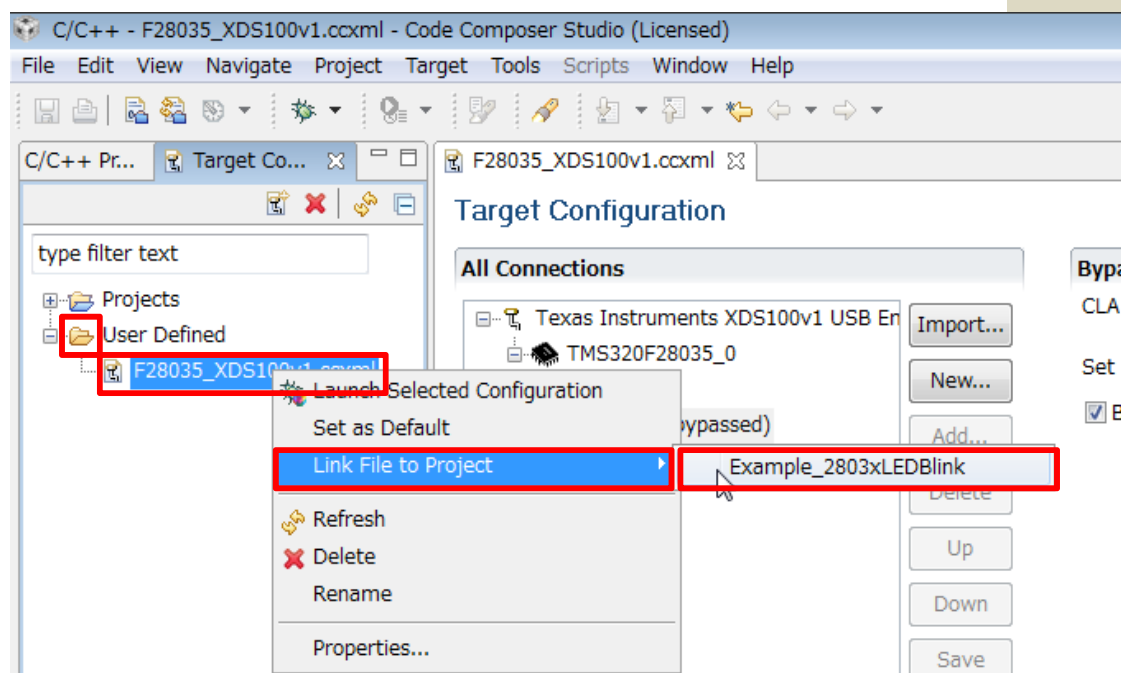
Open Perspectiveの切り替え

「Import Existing CSS/CCE Eclipse Project」が選択できない場合、ウィンドウが[Debug]の状態になっています。画面右上のボタンをクリックして、[C/C++]を選択してください。





[View]→[Target Configurations]を選択



[User Defined]の[+]をクリック

[F28035_XDS100v1]を右クリック

[Link File to Project]→[Example_2803xLEDBlink_]を選択

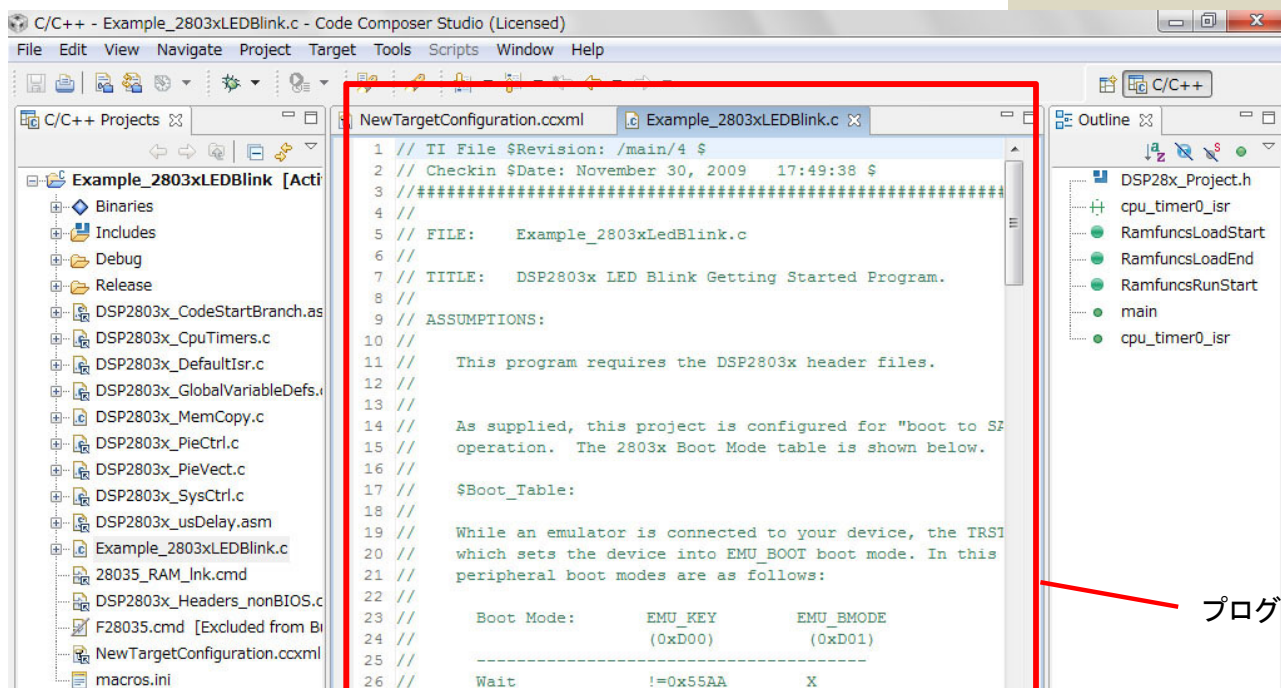
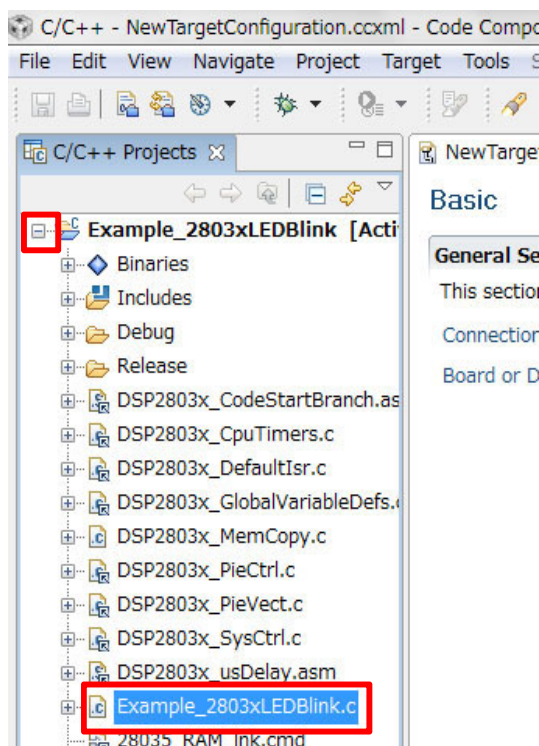
これでマイコンとプロジェクトと関連付けられ、本と伝票がそろいました。

⑤ プログラムの書き換え

いよいよ、プログラムをマイコンに分かるように翻訳(ビルド)して、通信(ロード)で送り届けます。その前に、Example_2803xLEDBlink.cのプログラムを書き換えてみましょう。

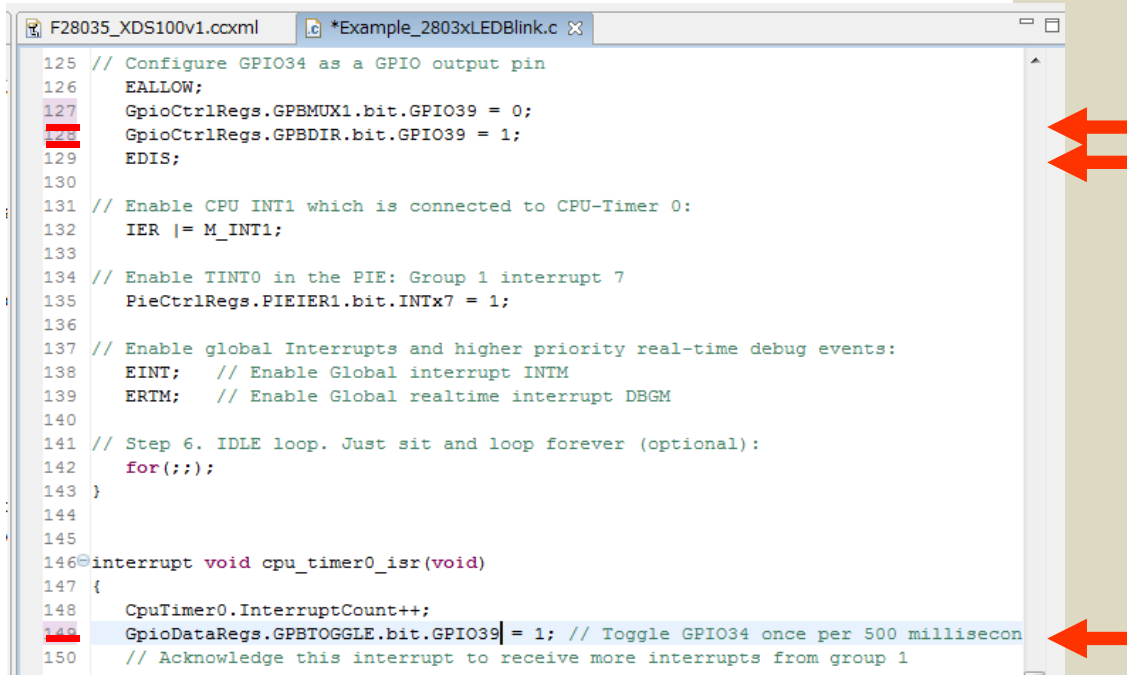
「Example_2803xLEDBlink.」の[+]をクリックして[-]にします。

「Example_2803xLEDBlink.c」をダブルクリックすると、右側にプログラム画面が表示されます。



プログラム画面

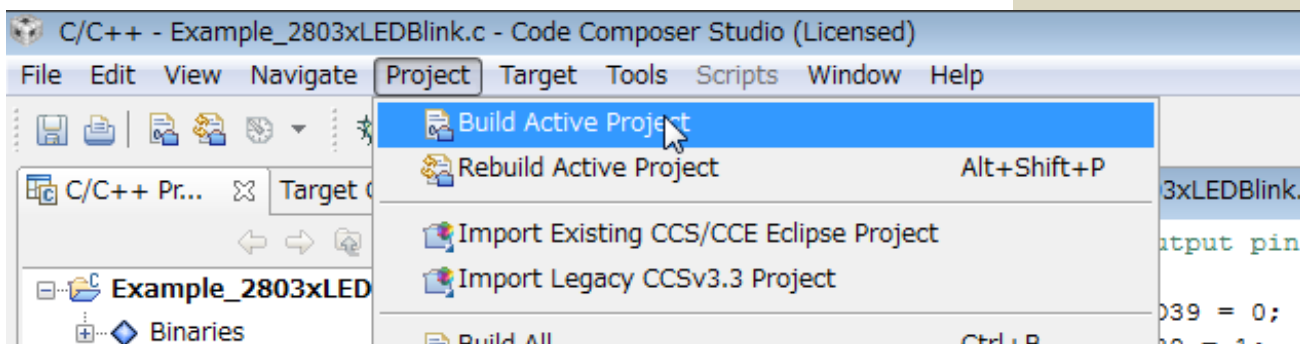
127,128,149行の「bit.GPIO34」を「bit.GPIO39」に数字を書き換えます。



```
125 // Configure GPIO34 as a GPIO output pin
126 EALLOW;
127 GpioCtrlRegs.GPBMUX1.bit.GPIO39 = 0;
128 GpioCtrlRegs.GPBDIR.bit.GPIO39 = 1;
129 EDIS;
130
131 // Enable CPU INT1 which is connected to CPU-Timer 0:
132 IER |= M_INT1;
133
134 // Enable TINT0 in the PIE: Group 1 interrupt 7
135 PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
136
137 // Enable global Interrupts and higher priority real-time debug events:
138 EINT; // Enable Global interrupt INTM
139 ERTM; // Enable Global realtime interrupt DBGM
140
141 // Step 6. IDLE loop. Just sit and loop forever (optional):
142 for(;;);
143 }
144
145
146 interrupt void cpu_timer0_isr(void)
147 {
148     CpuTimer0.InterruptCount++;
149     GpioDataRegs.GPBTOGGLE.bit.GPIO39 = 1; // Toggle GPIO34 once per 500 millisecond
150     // Acknowledge this interrupt to receive more interrupts from group 1
151 }
```

⑥ ロボット語に翻訳、ファイルをつくる

プログラムの変更を終えたら、「ビルド」(*54)します。



[Project] → [Build Active Project]

これで、プロジェクトのビルドは完了です。

⑦ マイコンとコンピュータをつなぐ

ビルドされたプロジェクトをマイコンに書き込みます。USBケーブルでマイコンとコンピュータをつないでください。

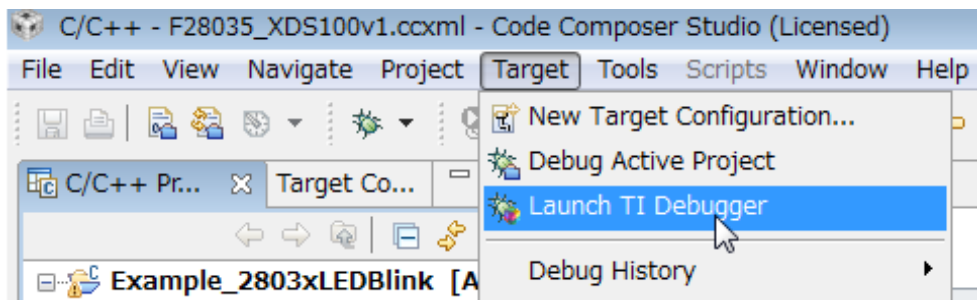


(*54)
ビルドとは

沢山のプログラムをコンパイル(コンピュータ言語に翻訳)して、マイコンに渡すための一冊の本にすることです。

⑧ プログラムの間違い探し

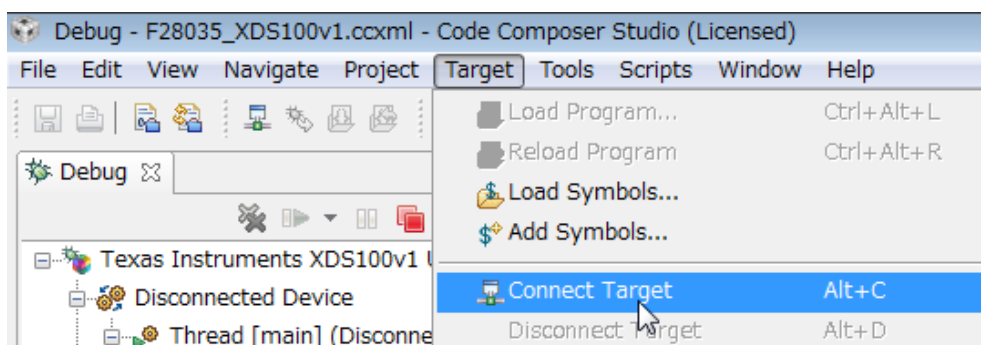
[Target]→[Launch TI Debugger]を選択して、デバッガ(*55)を起動します。(*56)



⑨ プログラムを送る

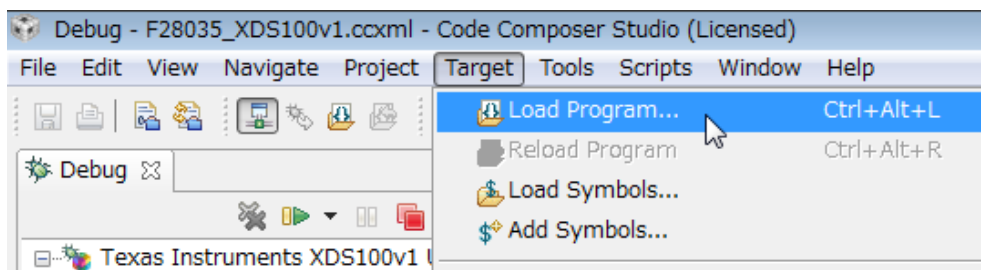
コンピュータにつないだマイコンを認識させます。

[Target]→[Connect Target]



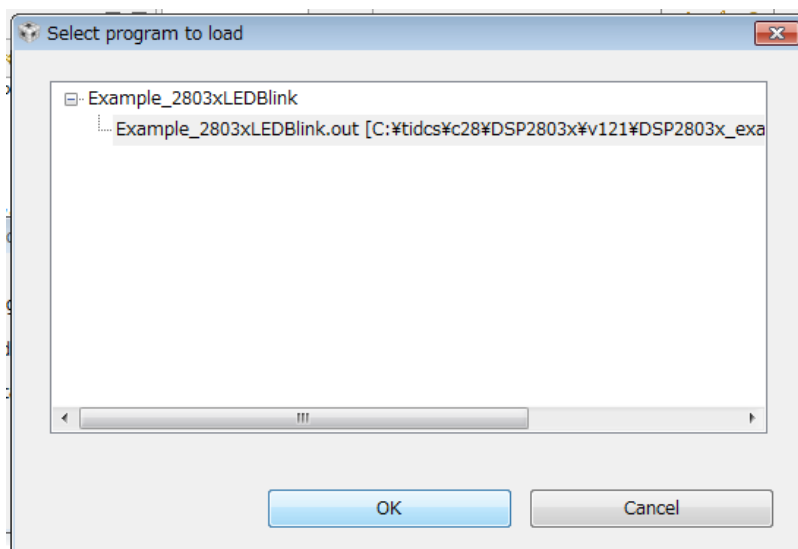
ビルドしたファイルをマイコンに送り込みます。

[Target]→[Load Program]



Debugフォルダに入っている.outファイルを指定します。

[Browse project...]をクリックし、下記画面のファイルを選択します。



(*55)
デバッガとは

「バグ(bug)」とは虫のこと。プログラムの誤りや欠陥のことをさします。もちろん本物の虫ではありませんが、真空管やリレーなどの電気部品が使われ大型だった昔の計算機では、内部に本物の虫が入り込んで故障することもあったようです。

「デバッグ(Debug)」とは、バグを探して修正することをいいます。正しく、安定したプログラムを作るにはデバッグが必要不可欠です。

「デバッガ」とは、バグ取りツールのことです。プログラムを書くとき必ずミスをするもので、そのミスがバグとなります。大きなプログラムになるとバグを探し出すのは至難の技で、そのためデバッガという専用のツールが必要になります。非常に便利なものです。

(*56)
トラブルシューティング

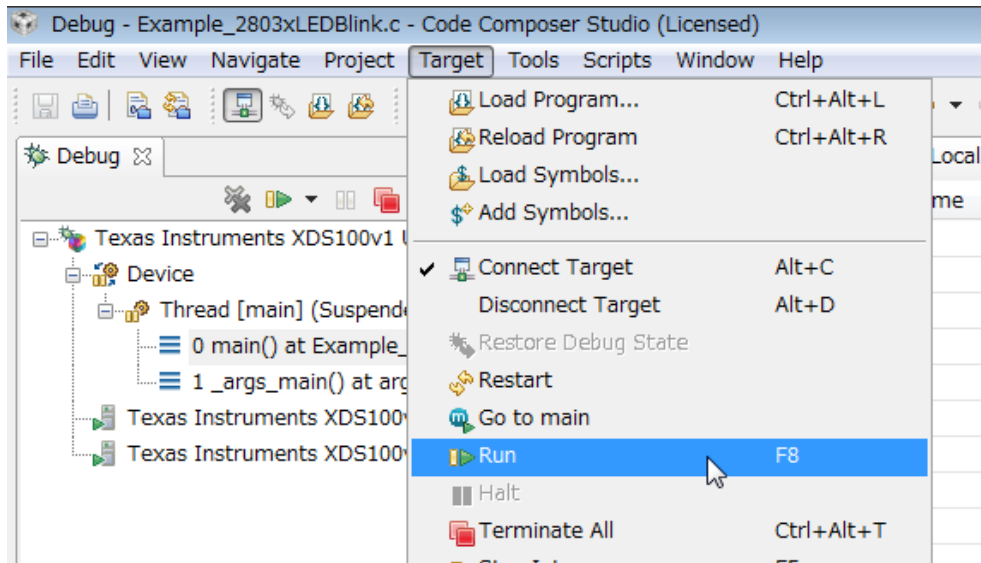
ひょっとしたら⑧の後にエラーが発生するかもしれません。その時はCCSを一度閉じてから、

⑦「マイコンとコンピュータをつなぐ」

を先に行ってからCCSを立ち上げなおして作業をやり直してみてください。

⑩ 動かす

いよいよ、プログラムを実行します。[Target] → [Run]

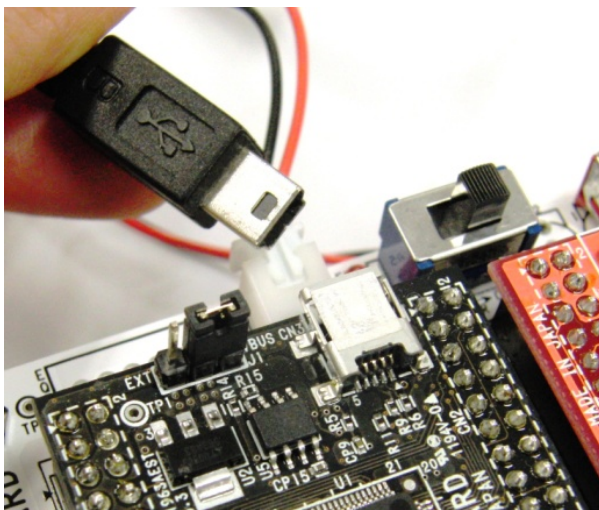


これで実行できたはずですが。

マイコンボード上のLED1が点灯するはずですが。できましたか？

⑪ とりはずす

[Target] → [Disconnect] で自由にケーブルをはずせます。



まとめ

開発環境をインストールして、プログラムの書き込みを覚えたら、ここから使いこなすためには、とにかく数が勝負です。次章からは、いろいろなサンプルプログラムを動かしてみます。

プログラムを書いたら、とにかく次の3つのステップを繰り返します。

[Project] → [Build Active Project] ファイルをコンパイルしてメモにする

[Target] → [Load Program] そのメモを読み込ませる

[Target] → [Run] 実行

「ビルド」「ロード」「実行」です。そうしないといくら頑張ってもマイコンは「どこ吹く風」状態です。

ロボットデビュー、おめでとうございます！

4-4 Arduino互換の開発環境「f-palette IDE」のインストール

さてここからは、もう一つの開発環境について解説します(*57)。世界中のホビーストに愛されている「Arduino」(*58)とほぼ同じ開発環境(IDE: Integrated Development Environment)で、f-paletteを動かすことが可能です。ここではCCS4から開発環境を移行し、プログラム開発を行うための準備を完了するところまでの解説を行います。

- ① ブートローダのインストール(開発環境の変更)
- ② 開発環境のインストール
- ③ 動作確認

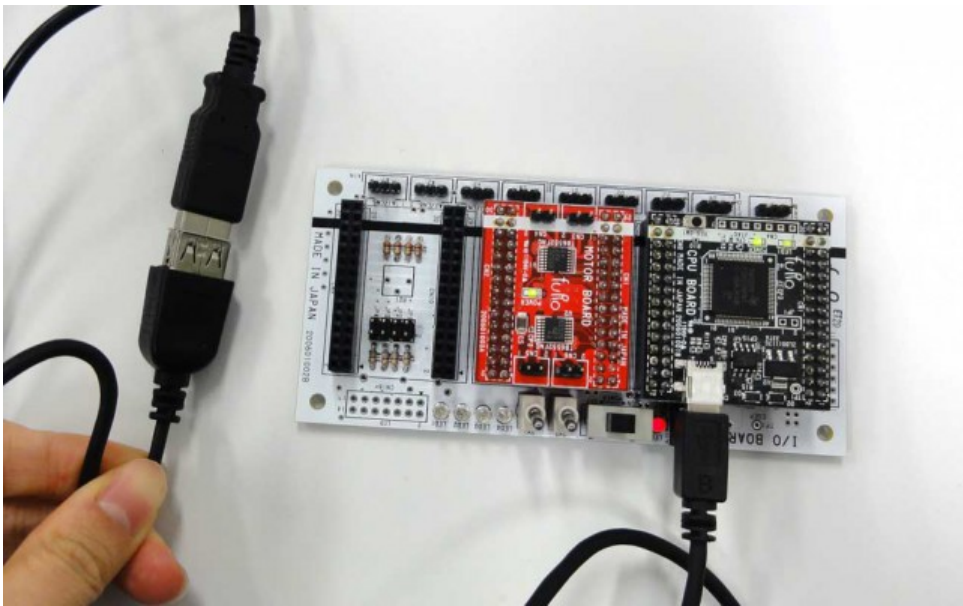
① ブートローダのインストール(開発環境の変更)

Arduinoの開発環境(f-palette IDE)を使うためには、既にパソコンにCCS v4がインストールされている必要があります。その上で、CCS v4から開発環境を変更します。まだの人は、4-1に戻って、まずCCSのインストールから始めてください。

それでは、以下の手順に沿って作業を行ってください。

手順1 f-paletteをつなぐ

f-paletteをUSBケーブルでパソコンとつなぎます。



手順2 ブートローダのダウンロード

F-palette IDEを立ち上げる前に、マイコン上でブートローダが起動している必要があります。ここではCCSを使ってブートローダを書き込みます。

ブートローダ「ccs4.zip」は下記のサイトからダウンロードできます。

<http://www.f-palette.org/started/1076/>

ダウンロードをしたら、適当なフォルダに解凍します(ここでは説明のためC:\project\fpaletteに解凍)。

(*57)

全ての人が行う必要はありません

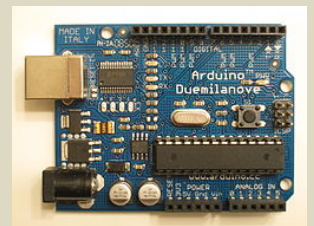
「俺はCCS1本で行く」という一途な人は読み飛ばしても構いませんし、しばらくCCSを使った後に、そういえば別の開発環境があったな、と思い出してから戻って来るのもいいでしょう。

(*58)

Arduinoとは

アルドウイーノと読みます。生まれはイタリアです。電子工作が必ずしも得意でないデザイナーやアーティストでも、簡単にメディアアート作品などのプロトタイプを制作できることを目的とした、マイコン搭載ボード、工作キット及び開発環境のこと。ここでいうArduino互換とは、開発環境「Arduino IDE」とほぼ同じ環境を用意しているということ。

ちなみに、ArduinoではIDEを使ってプログラム開発することを「スケッチ」と言います。



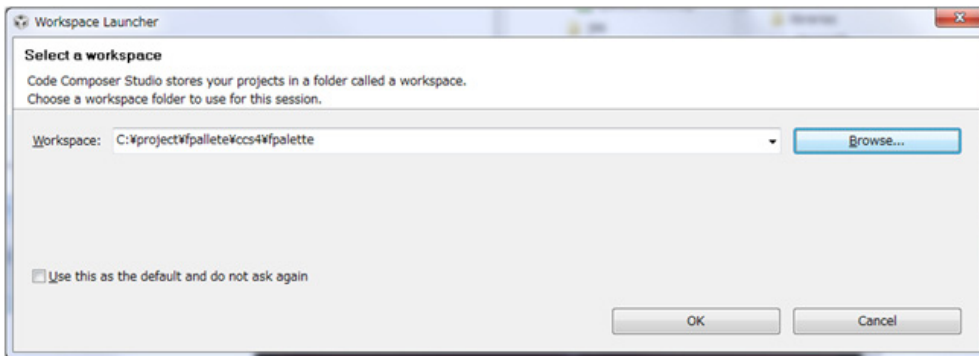
Arduinoボード

手順3 CCSでブートローダを書き込み

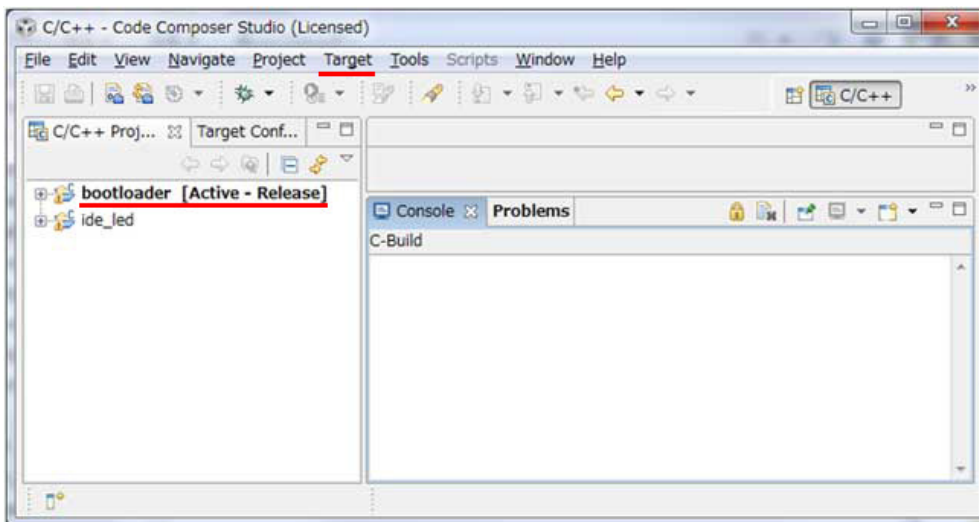
次に「Code Composer Studio v.4」を立ち上げます。ワークスペースは、

C:\project\palette\ccs4\palette

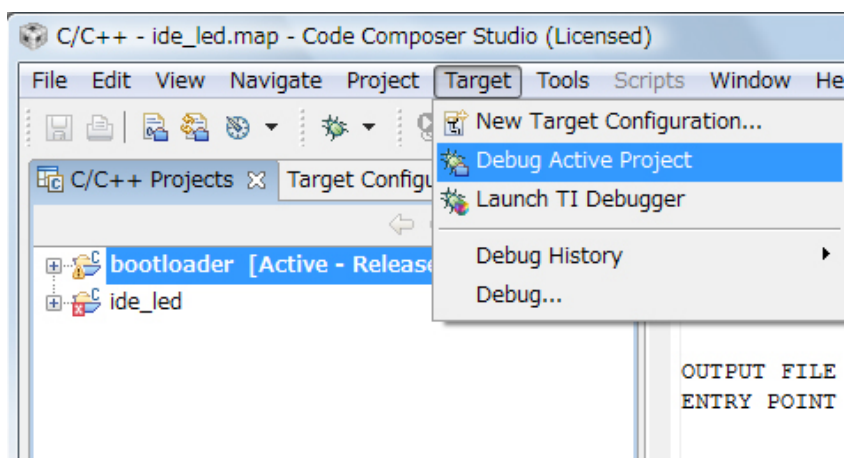
を指定して『OK』をクリック。



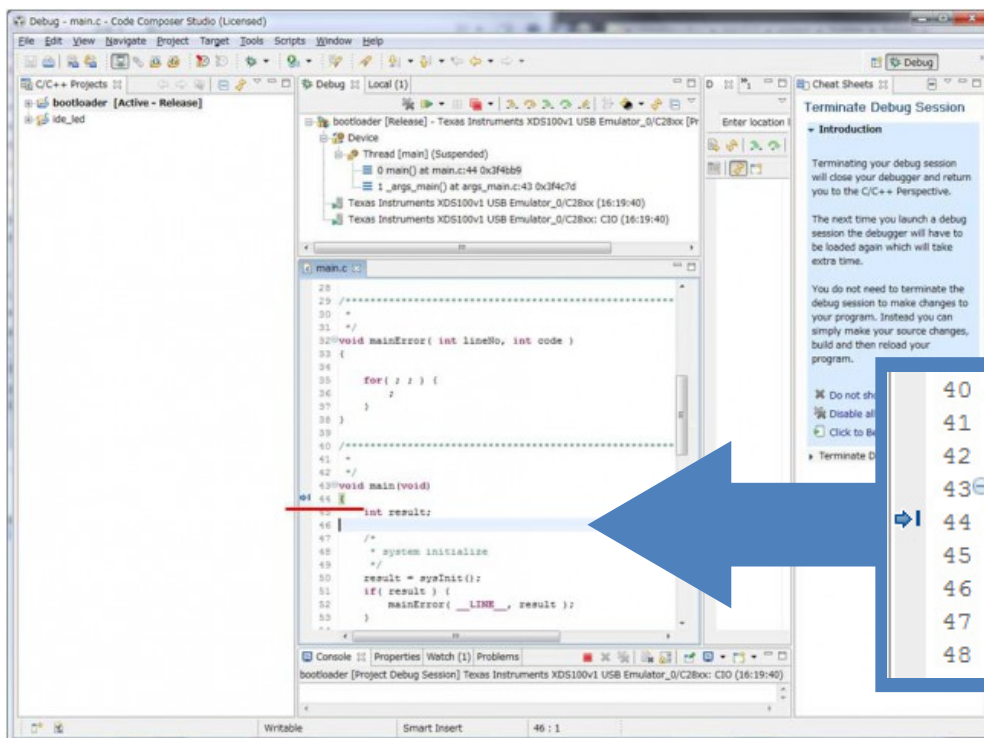
bootloader [Active - Release] となっていることを確認して、



「Target」→「Debug Active Project」でターゲットにブートローダーを書き込みます。



しばらく待つて・・・正常に完了すると、デバッガが起動して、main(void)で止まった状態になります。



(*59) ブートローダの動作

マイコンに書き込まれているブートローダは、起動時に5秒間シリアル通信を監視しています。この間にホストPCから書き込みが開始されない場合、すで書き込まれているプログラムを起動します。この間、基板上のLEDが点灯します。

また、ブートローダ書き込み時にすべてのフラッシュメモリが消去されてしまうため、1度も書き込みが行われていない場合は、LEDが暗くなります。

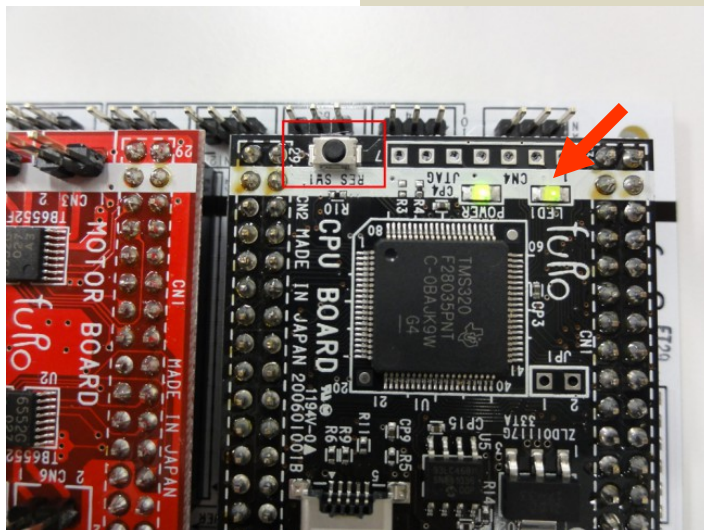
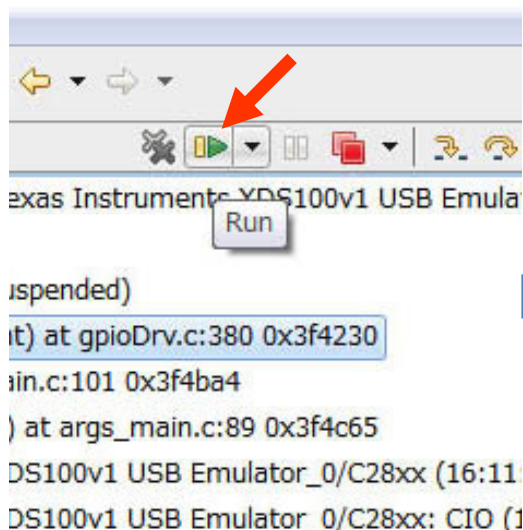
```
40  /******  
41  *  
42  */  
43 void main(void)  
44 {  
45     int result;  
46  
47     /*  
48     * system initialize
```

停止画面例

手順4 動作確認とCCSの終了

次に、ブートローダがちゃんとインストールされていることを確認します。CCSのツールバーにある「Run」ボタンを押すと、CPU基板上の「LED1」が点灯し、約5秒後にLEDが少し暗くなります。これが確認できたなら、正常にインストールされています。もし動かなかつたら、最初からやり直してください。

(*59)



そのままCCSを終了します。右上の「×」を押すか、「File」→「Exit」。
これで、ブートローダのインストール(開発環境の変更手続き)は終了です。

② 開発環境のインストール

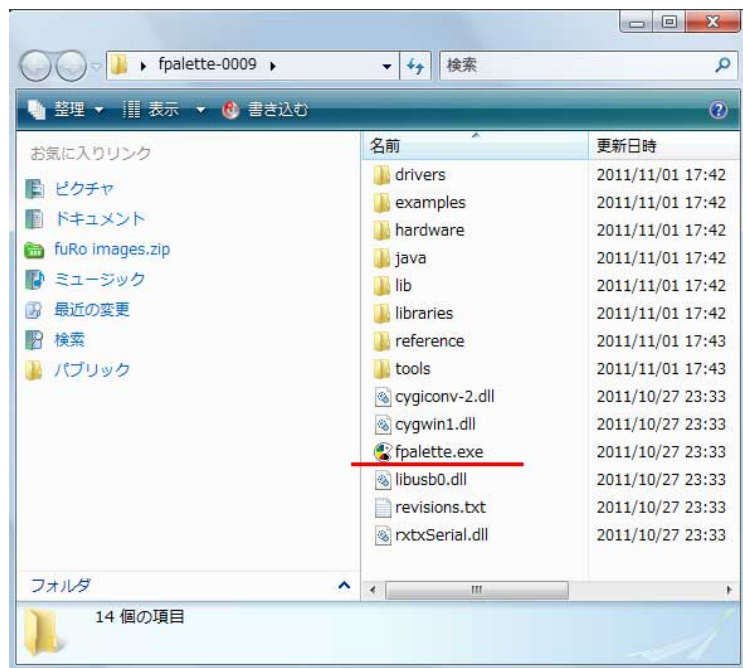
次は新しい開発環境をインストールしましょう。開発環境「fpalette-0009.zip」は下記のサイトからダウンロードできます。

<http://www.f-palette.org/started/2935/>

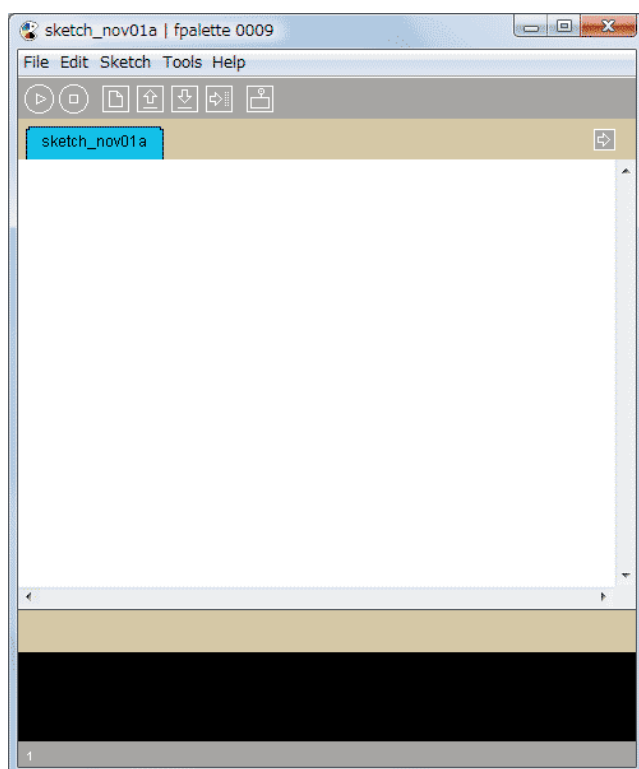
ダウンロードをしたら、任意の場所で解凍してください。これで終了です。

③ 開発環境の起動

「fpalette.exe」をダブルクリックすると起動します。

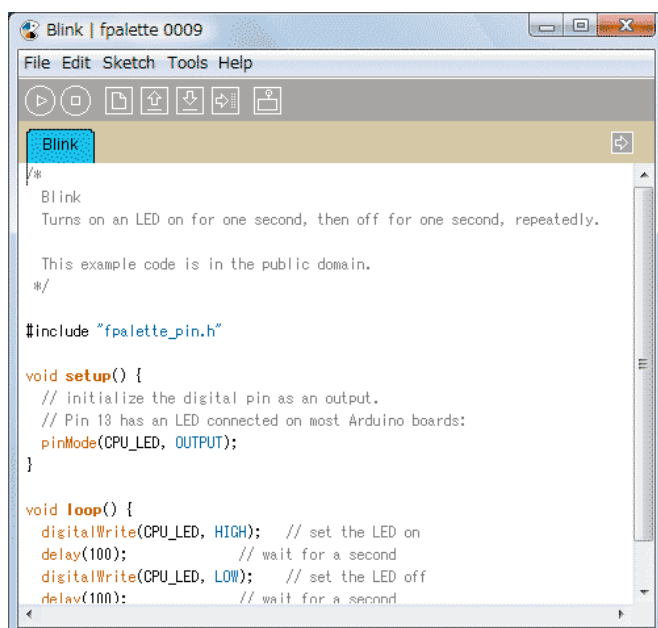
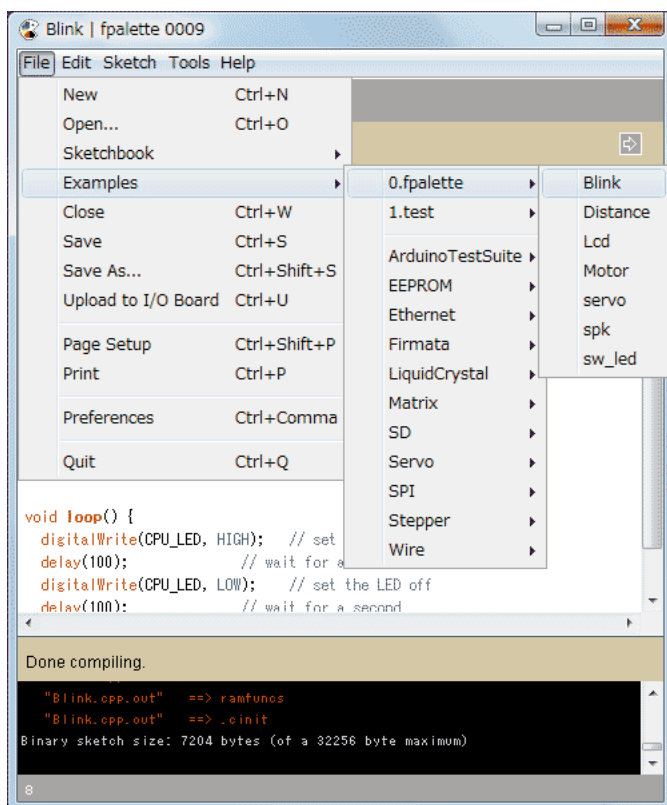
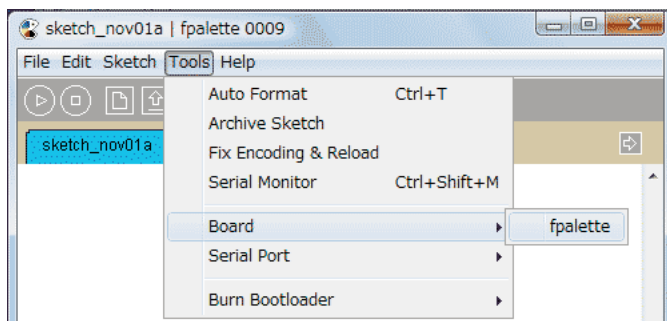


起動画面。



④ 動作確認

開発環境が正常にインストールされ、プログラムをロードしてちゃんと動かすことができるか、サンプルプログラムを使って確認します。



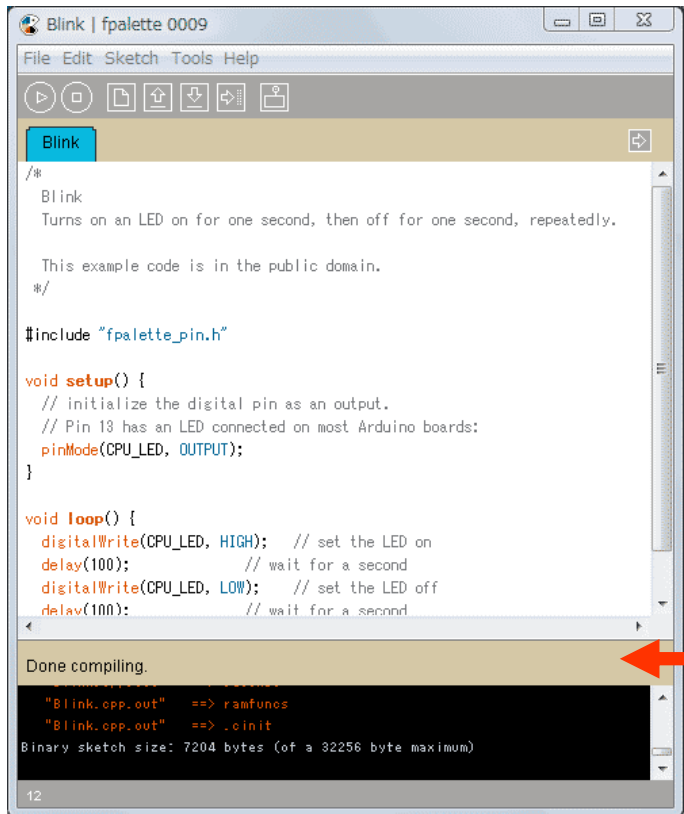
f-paletteは以下の流れで、マイコンにプログラムを書き込みます。(*58)

- 1) プログラムを呼び出す
- 2) プログラムの確認「Verify」
- 3) チップにプログラムを書き込む「Upload」
- 4) 実行「リセットボタンを押す」

「Tools」→「Board」→「fpalette」を選択します。

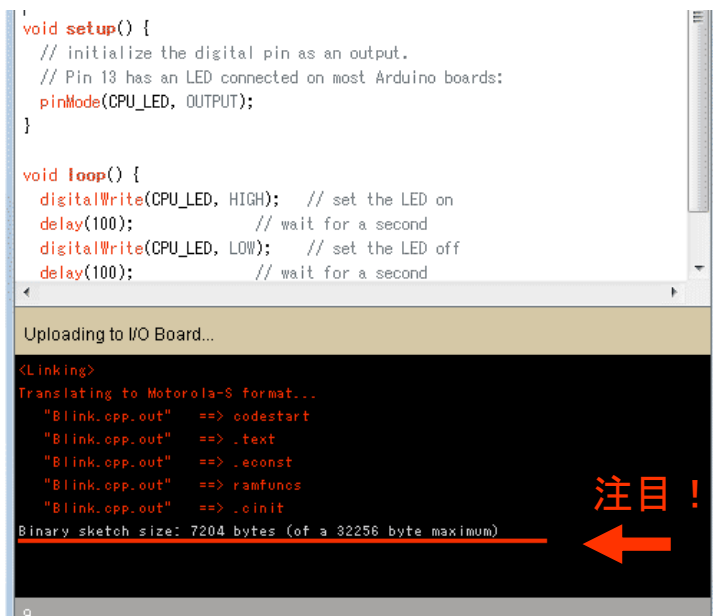
「File」→「Example」→「0.fpalette」→「Blink」を開きます。このとき、新しいWindowが開きます。不要なWindowは閉じてかまいません。

サンプルプログラム「Blink」が読み込まれました。



「Sketch」→「Verify/Compile」でビルドの確認をします。

Done compiling.となっていたら、プログラムが正しいということです。プログラムにエラーがあると、それが表示されます。



「Tools」→「Serial Port」→「COMx」で使用するシリアルポートを選択します。

「File」→「Upload to I/O Board」でビルドを行った後、マイコンへの書き込みが行われます。その時に、画面下部のIDEからのメッセージが表示される場所に注目してください。

Binary sketch size: **** bytes (of a 32256 byte maximum)

と表示されたら、**おおよそ15秒以内**に、CPUボード上のリセットボタンを1回だけ押下してください(あるいはCPUボードの電源を投入してください)。これを行わないと書き込みが行われません。

チカチカ！

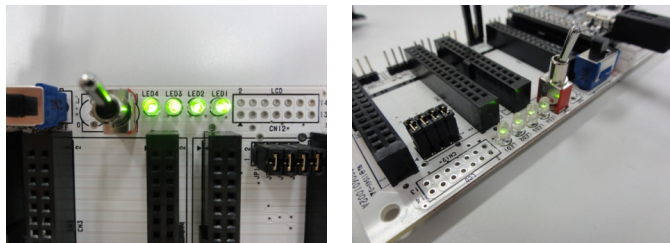


ちょっとだけ待って・・・CPUボード上の「LED1」が点滅を始めたら、書き込みテストは正常に完了です。プログラム開発の準備はこれで整いました。

5章 LEDをぴかぴかさせてみようーデジタル入出力(DIO)機能を使う

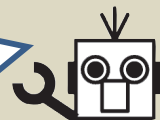
概要

マイコンの得意技である「スイッチON/OFF機能」を使って、どんなことができるのでしょうか。まずはその中の基本、「デジタル入出力(DIO)機能」を使って、I/OボードのLEDを点滅させます。次に、I/Oボードに取り付けたトグルスイッチの切り替えを読み込んで、LEDの点滅の様子を変化させます。



- 5-1 サンプルプログラム『LED_Switch』の実行
- 5-2 プログラムの中身を見てみよう
- 5-3 LED_Switchチャレンジ
- 5-4 f-palette IDEのプログラムの実行
- 5-5 デジタル入出力(DIO)機能とは

なにはともあれ、「やってみる」がモットーです。
何も考えずレッツトライ！！



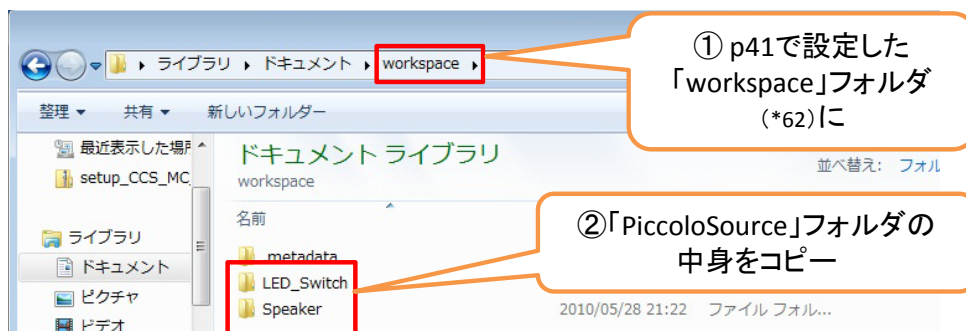
5-1 サンプルプログラム『LED_Switch』の実行

配布されているサンプルプログラムを、4-3で学んだのと同じ手順でインポートし、Build→Load→Runを行ってみましょう。(*60)

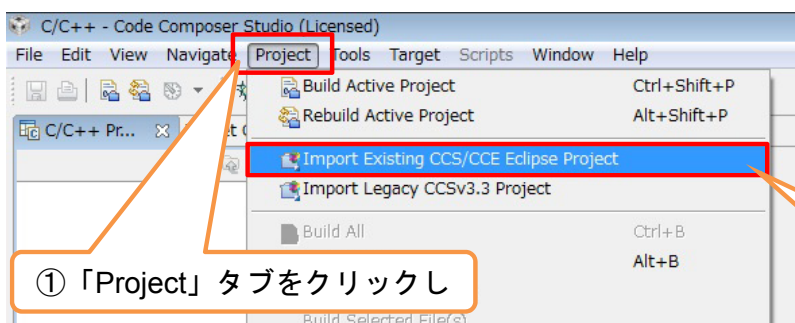
5-1-1 プロジェクト(サンプルプログラム)のインポート

手順1 サンプルプログラムが収められている「PiccoloSource」フォルダ(*61)を、自分のPCの好きな場所へコピーします。

手順2 「PiccoloSource」フォルダの中身を、p41で設定した「workspace」フォルダの中にコピーします。



手順3 デスクトップのアイコンからCCSを起動し、上部の「Project」タブをクリックして、「Import Existing CCS/CCE Eclipse Project」を選択します。



(*60)

3つの手順をマスターせよ！

図4-3-1を思い起こしてください。この手順はこれから何度も行います。しっかり覚えましょう。忘れたら必ず4-3か、ここを見直してください。

(*61)

PiccoloSourceのダウンロード

PiccoloSourceのzip(圧縮)ファイルは下記のサイトからダウンロードできます。
<http://www.f-palette.org/samplecode/PiccoloSource.zip>
ダウンロード後、解凍しておきます。

(*62)

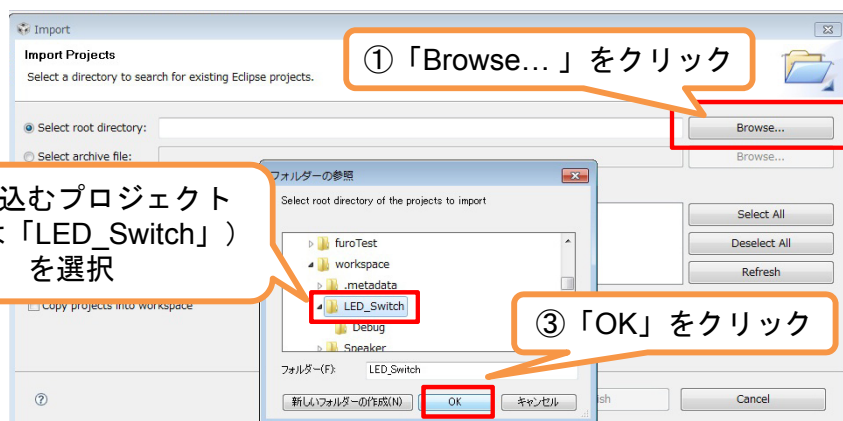
Workspaceの位置

「workspace」フォルダは、特に設定を変更していなければ、「マイドキュメント」の中にあるはずです。

C/C++perspective
(パースペクティブ)
に変更しないとこの
メニューは現れません

②「Import Existing
CCS/CCE Eclipse
Project」をクリック

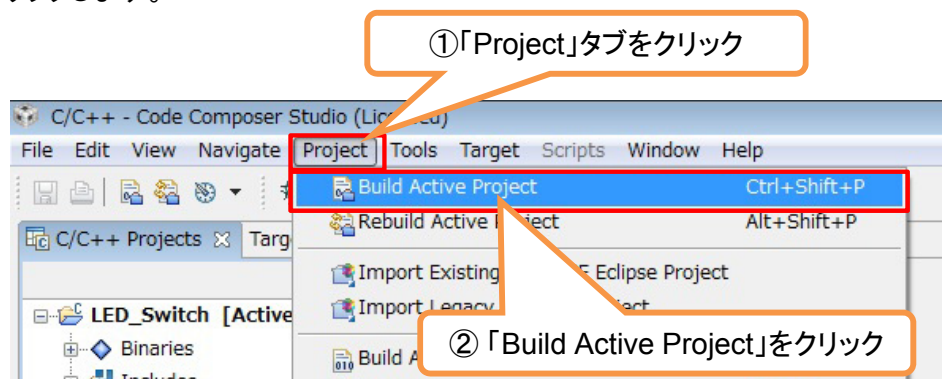
手順4 Importウィンドウが出たら、「Browse...」をクリックして、「workspace」にコピーした「LED_Switch」フォルダを選択した状態で「OK」をクリック。



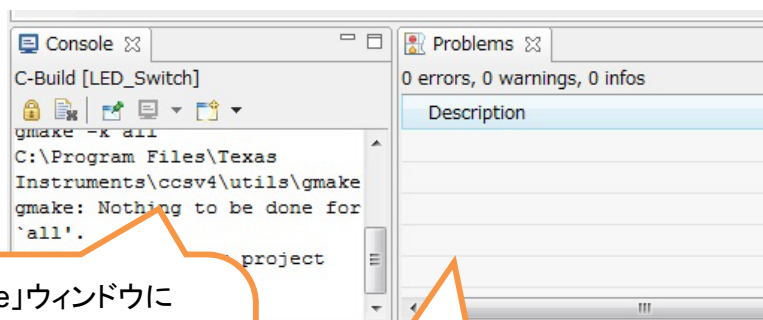
手順5 「Copy projects into workspace」にチェックを入れ「Finish」をクリックすると、選択したプロジェクト(サンプルプログラム)がインポートされます。

5-1-2 プロジェクトのビルド(*63)

手順1 画面上部にある「Project」タブをクリックし、「Build Active Project」をクリックします。



手順2 ビルドの状況は画面下の「Console」ウィンドウに表示され、もしエラーなどがあった場合は「Problems」ウィンドウに表示されます。

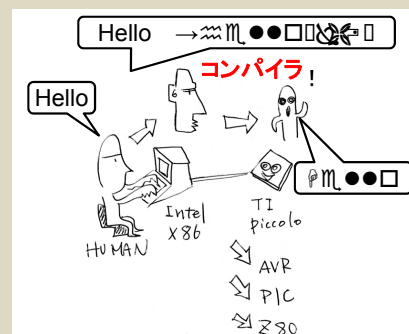


①「Console」ウィンドウに
‘Finished building target:
(プロジェクト名).out’
‘
Build complete for project
(プロジェクト名)
と表示され、

②「Problems」ウィンドウに赤色の
× 印で「Error」が表示されて
なければOKです。

(*63)
ビルドとコンパイル

コンパイルは3-5で説明した、人間の言葉をマイコンが分かるように翻訳する作業のことです。



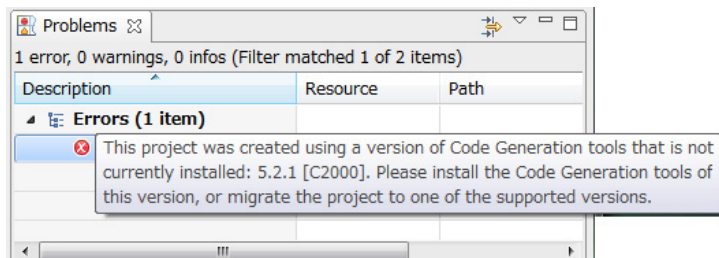
ビルドはたくさんのファイルをコンパイルし、一つのまとまりにしてマイコンに渡しやすい形にする作業のことです。

手順3 もしもビルド時にエラーが出てしまったら・・・

ビルドの時点で、「Problems」ウィンドウに **×** が表示され、

This project was created using a version of Code Generation tools
that is not currently installed :5.2.1[c2000]

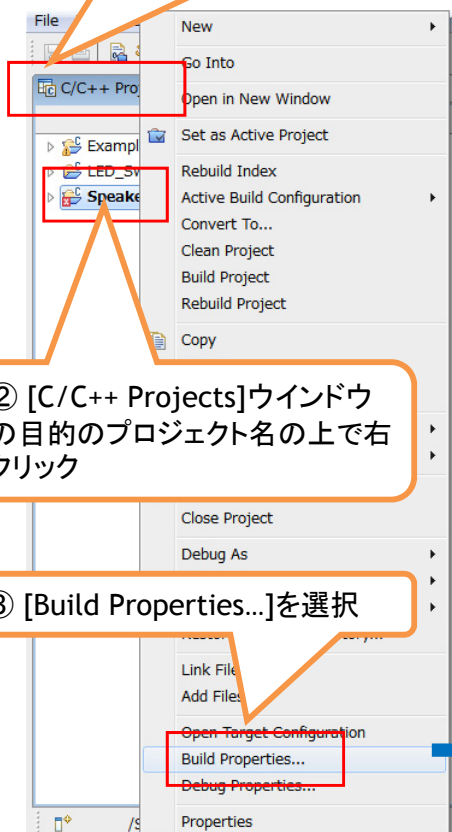
というエラーメッセージが出て、先に進めない場合があります。



このエラーは、サンプルソースを作ったときの環境と、今手元で使っている環境のバージョンが違うことから発生します。つまり、コンパイラのバージョンが違って、コンパイラのファイルのパスが変わっている事が原因です。ですから、プロジェクトに登録されているコンパイラ(ここであるCode Generation tools)を、インストールされているバージョンに変更することで解決します。

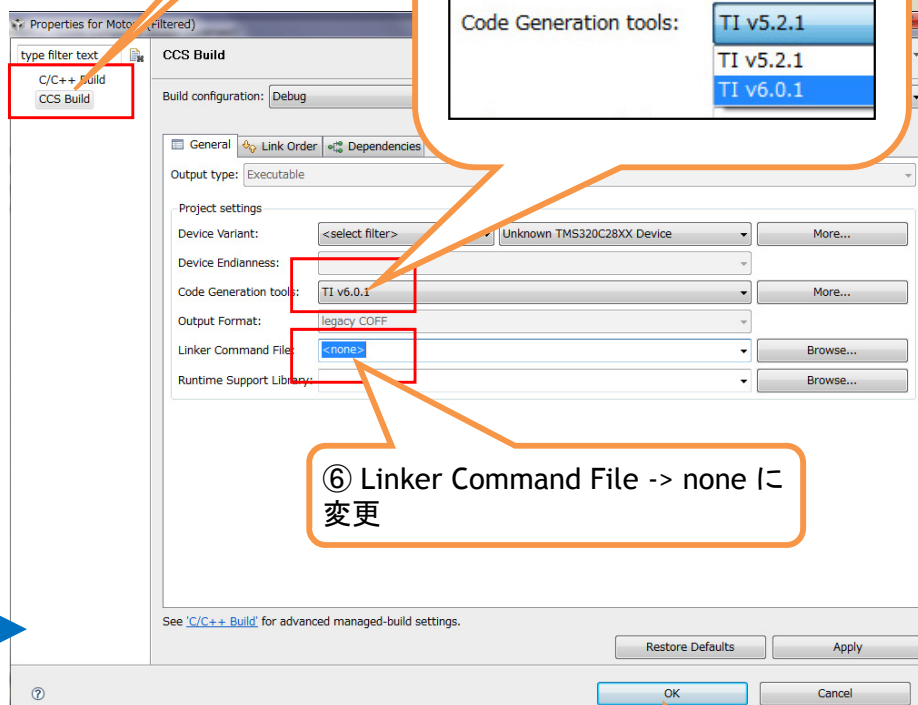
手順は次の通りです。

① 右上環境のモードが C/C++ Perspective になっているか確認(デバックモードでないということ)



② [C/C++ Projects]ウィンドウの目的のプロジェクト名の上で右クリック

③ [Build Properties...]を選択



⑤ [General Tab]中の Code Generation tools を TI v5.2.1からTI v.6.0.1など表示に出てくるものに変更

⑥ Linker Command File -> none に変更

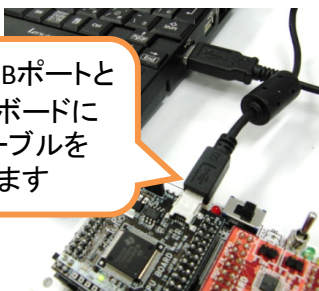
⑦ OKを選択

手順1に戻って、ビルドからやり直してください。エラーなくビルドできるようになっているはず。

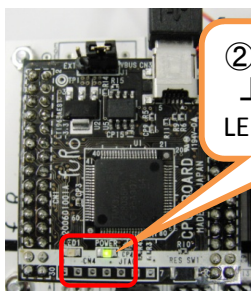
5-1-3 マイコンの接続(コネクト)

手順1 USBケーブルでPCとマイコンをつなぎます。

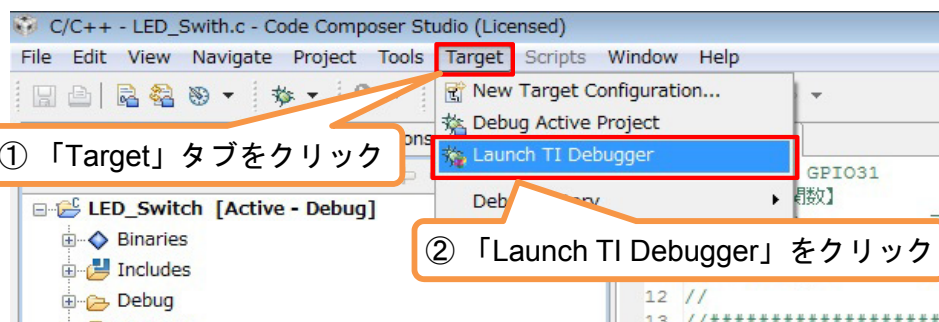
①PCのUSBポートとマイコンボードにUSBケーブルを挿します



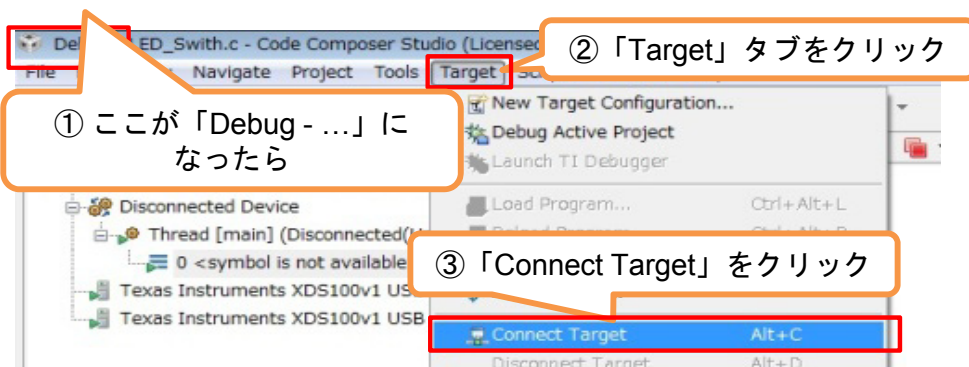
②マイコンボード上の「POWER」LEDが点灯します



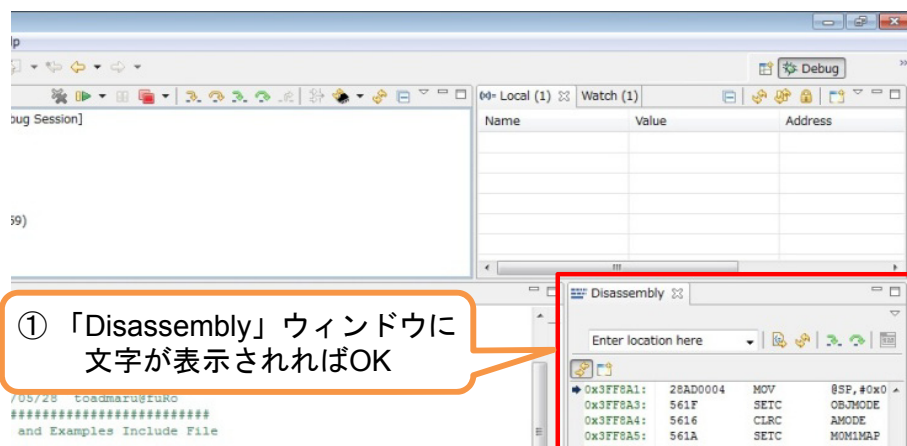
手順2 ビルドが終了したら、「Target」タブから「Launch TI Debugger」をクリックします。(*64)



手順3 ウィンドウ上の文字が「C/C++ - ...」から「Debug - ...」になったら、TI Debuggerが立ち上がっています。「Target」タブから「Connect Target」をクリックし、PCとマイコンの通信を確立します。(*65)



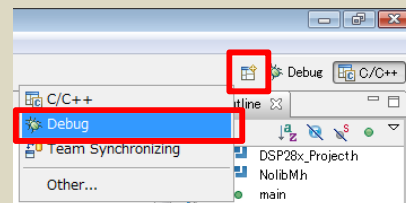
手順4 マイコンとの接続が確立すると、画面右下の「Disassembly」ウィンドウにマイコン内部のデータが表示されます。(*66)



(*64)

Open Perspectiveの切り替え

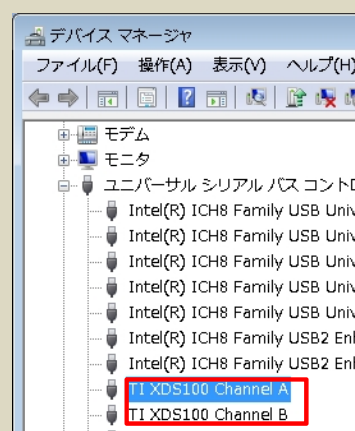
「Launch TI Debugger」が灰色になって選択できない場合、すでにDebuggerが立ち上がっています。画面右上のボタンをクリックし、「Debug」を選択してください。(*53)を参照のこと。



(*65)

トラブルシューティング

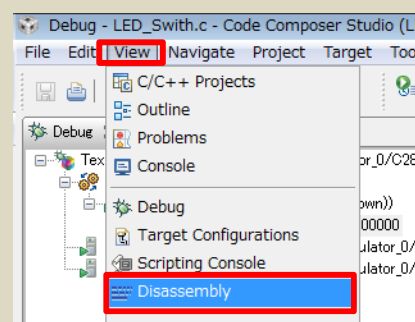
●Connectできない場合
→マイコンとPCはUSBケーブルで接続されていますか？
→Windowsのデバイスマネージャに「TI XDS100 Channel A/B」がありますか？
(ない場合はUSBケーブルを直し、CCSを終了し再度立ち上げてください)



(*66)

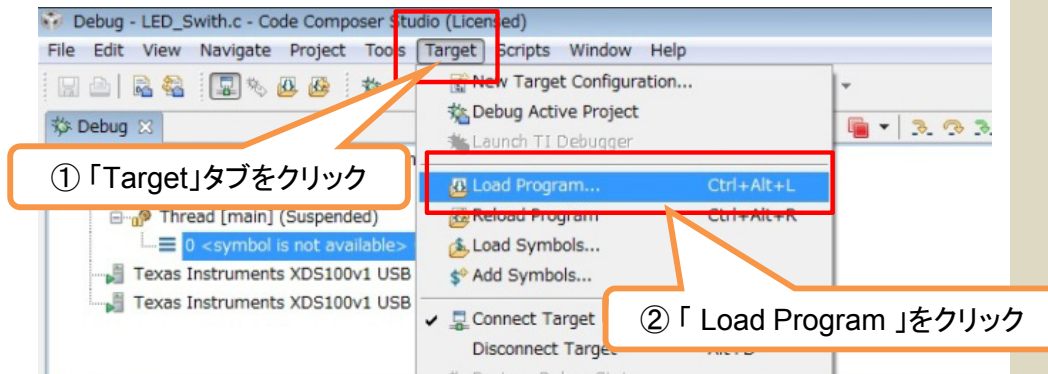
「Disassembly」ウィンドウの出し方

「Disassembly」ウィンドウがない場合は、画面上の「View」タブをクリックし、「Disassembly」をクリックしてください。

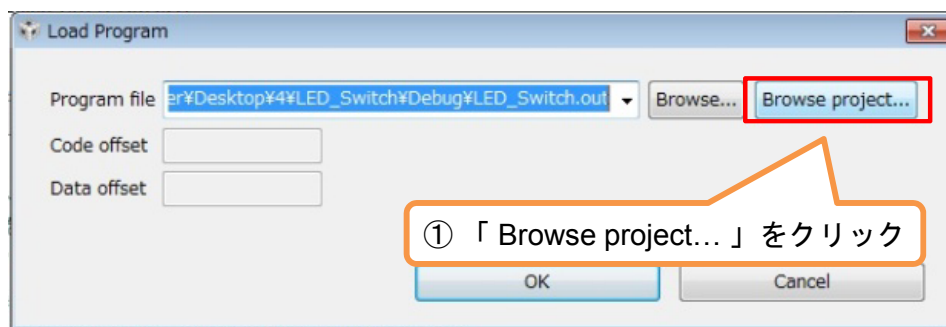


5-1-4 マイコンへのプロジェクトの書き込み(ロード)

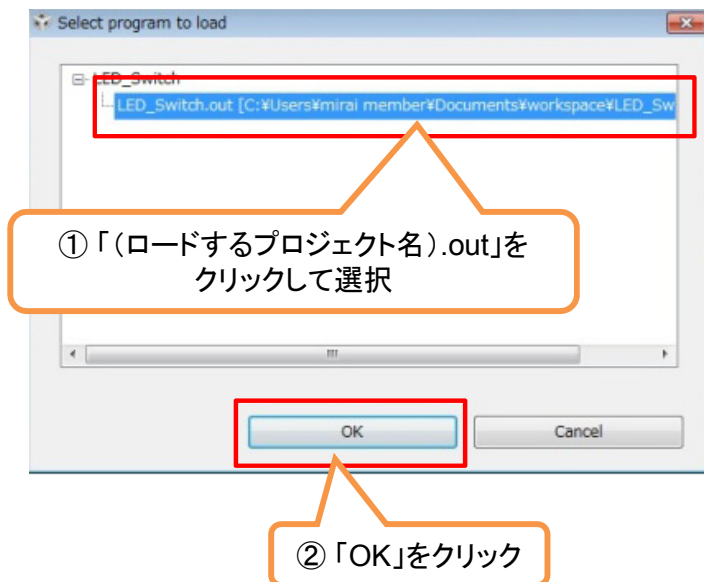
手順1 「Target」タブから「Load Program...」を選択します。



手順2 Load Programウィンドウが表示されたら、「Browse project...」を選択します。

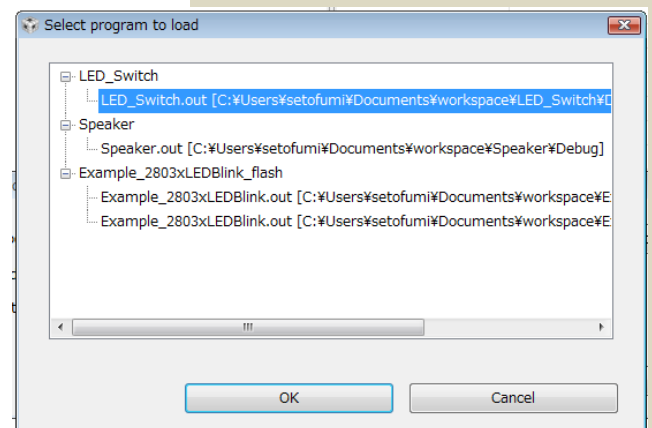


手順3 先ほどビルドし、今ロードするプログラム(ここでは「LED_Switch.out」)を選択した状態で「OK」をクリックします。(*67)

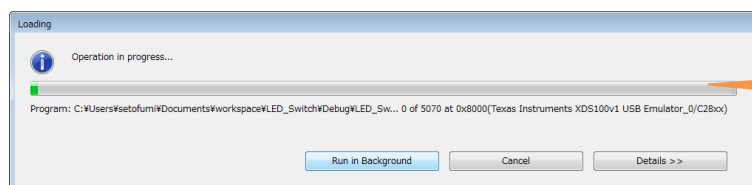


(*67)
ロードするプログラムの選択

「Select program to load」ウィンドウには、先ほどビルドしたプロジェクトだけではなく、これまでビルドしてまだ開いている他のプロジェクトも表示されます。ちゃんと目的のプログラムが選択されているか確認しましょう。



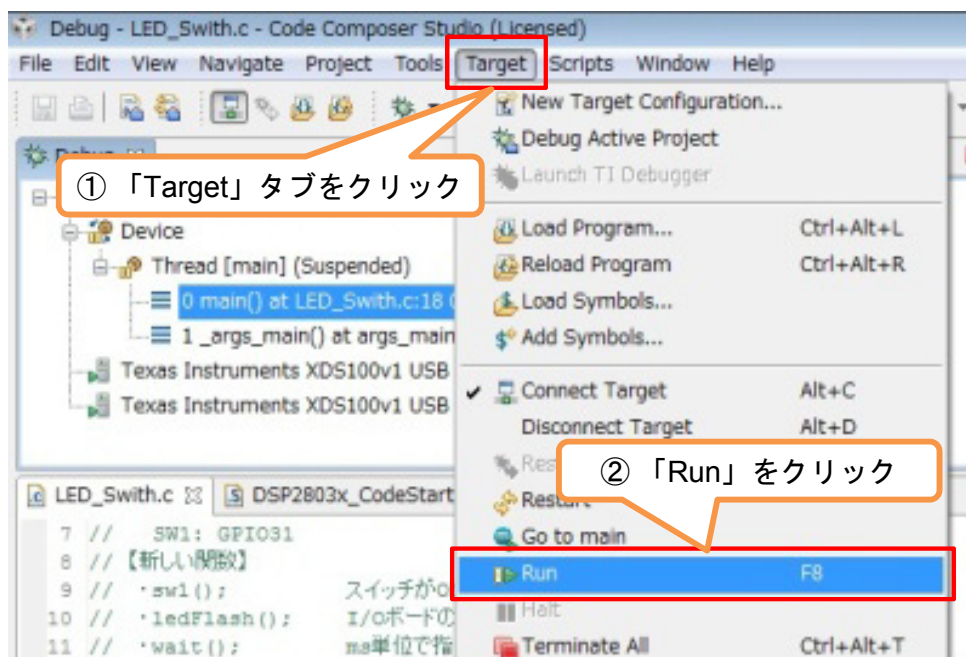
手順4 Load Programウィンドウに戻って「OK」をクリックすると、プログラムがマイコンに書きこめます。



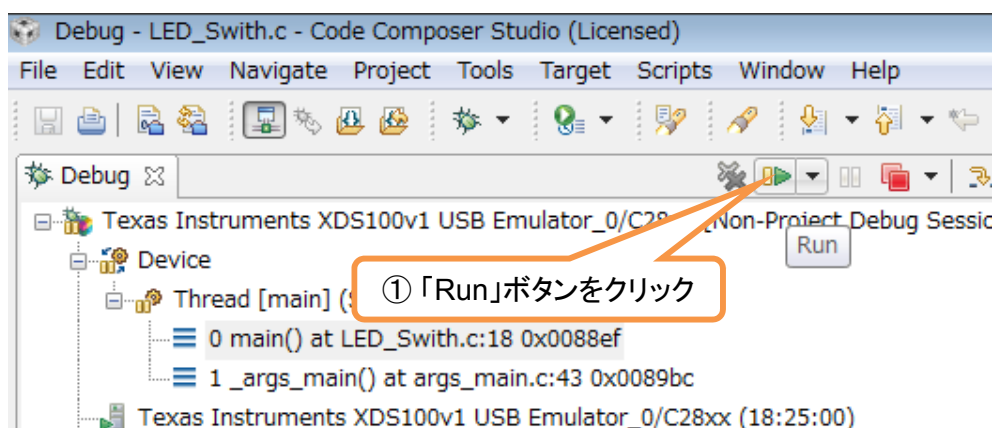
① ゲージが最後まで行ったら書き込み完了です

5-1-5 プログラムの実行(ラン)

手順1「Target」タブから「Run」を選択します。



もしくは、画面上にある緑の三角をクリックします。



IOボード上のLED1～4は点滅していますか？

SW1を切り替えたら、点滅の様子はどう変化しますか？ (*68)

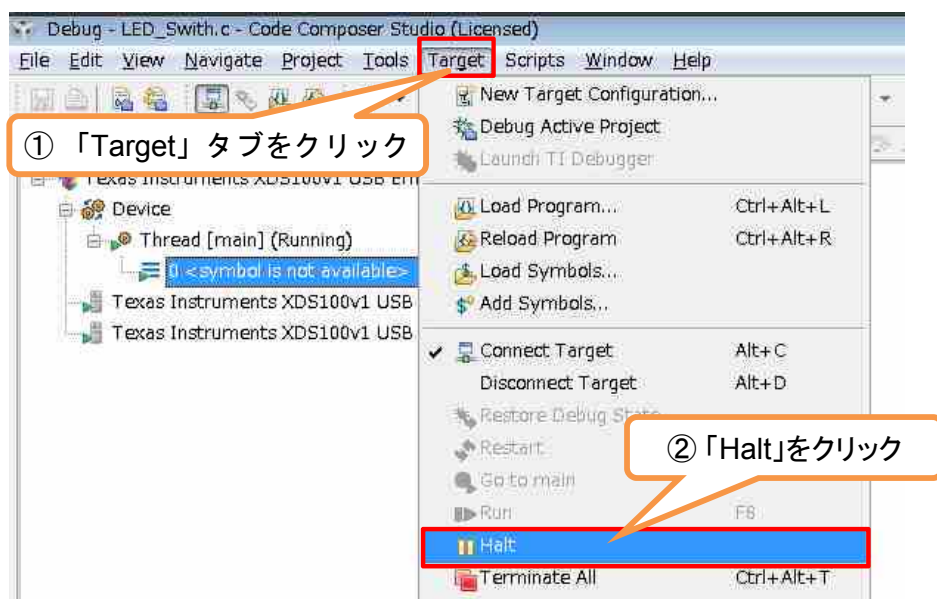
(*68)

トラブルシューティング

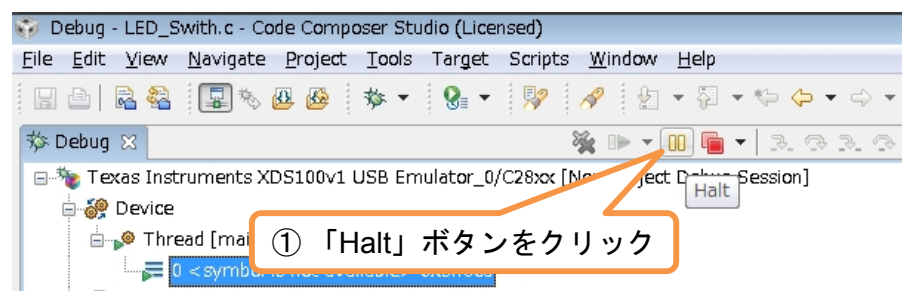
- LEDが全部点灯しない場合
→USBケーブルが接続されていますか？
- 一部のLEDが点灯しない場合
→R7, R9, R11, R12の抵抗器は正しくハンダ付けされていますか？
- SW1を切り替えても点滅の様子が変わらない場合
→IOボードにジャンパピンを7つ、正しく差し込んでいますか？
→R13, R14の抵抗器は正しくハンダ付けされていますか？

5-1-6 プログラムを停止する

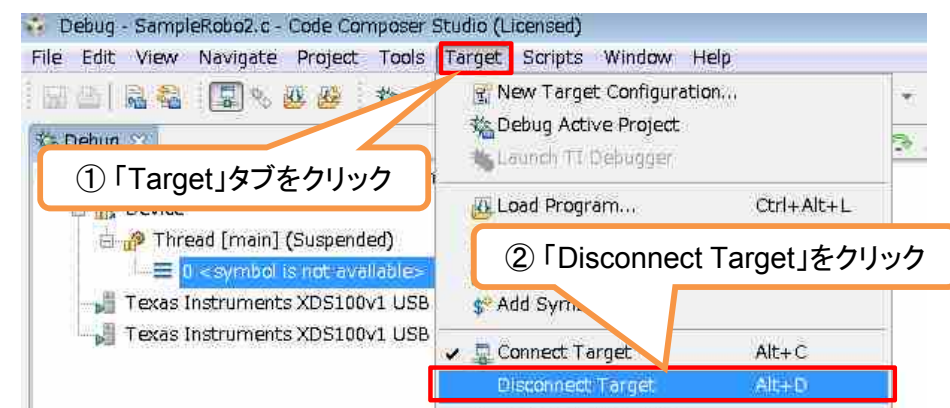
手順1 「Target」タブから「Halt」を選択します。(*69)



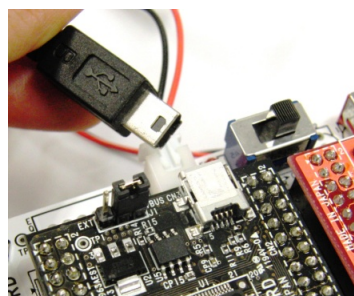
もしくは、画面上にある黄色の四角をクリックします。



手順2 「Target」タブから「Disconnect Target」を選択します。

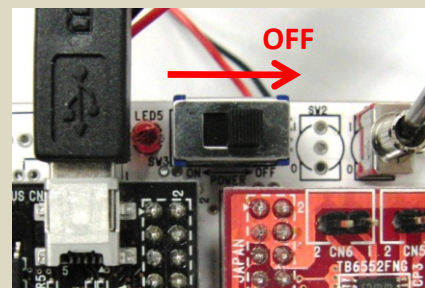


ここまですれば、USBケーブルを抜いても大丈夫です。



(*69)
I/Oボードの電源

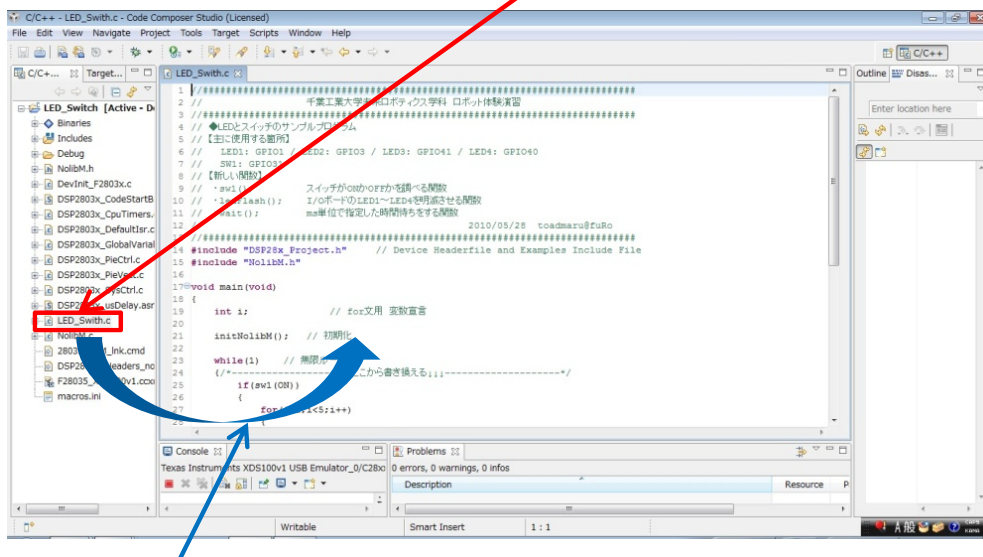
この後、I/Oボードの電源を入れて実行するサンプルプログラムも出てきます。その時には「Halt」を選択する前に、I/Oボードの電源をOFFにしましょう。



5-2 プログラムの中身を見てみよう

サンプルプログラム『LED_Switch』を実行して、LEDを点滅させ、スイッチの切り替えに応じてLEDの点滅を変化させることができました。それはプログラムでどうやって実現しているのでしょうか。「LED_Switch.c」を見てみましょう。

- ① LED_Switchプロジェクト内の「LED_Switch.c」をダブルクリックします。(*70)



- ② 右のウィンドウに「LED_Switch.c」の内容が表示されます。

- ③ main関数(void main(void)以降)の内部が、Projectをビルドして実行(Run)した際に実行されます。

条件分岐

if(条件){(処理1)} (*71)

◆ 条件が満たされたときにだけ、(処理1)を実行する

else{(処理2)}

◆ ifとセットで用いられ、ifの条件が満たされなかったときに(処理2)を実行する

38～62行目
while(1)による無限ループ

if(sw1(ON))

SW1が「ON」ならば
{ }の中(41～52行目)を実行

else

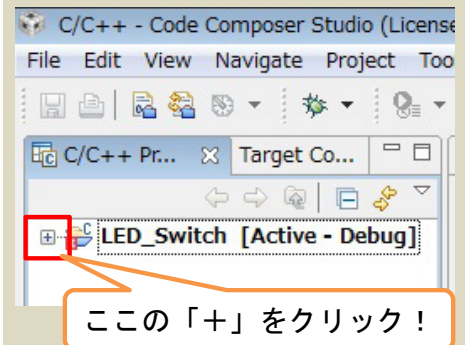
上記の場合でなければ
(SW1が「OFF」ならば)
{ }の中(53～61行目)を実行

```
38 while(1) // 無限ループ
39 { /*-----ここから
40     if (sw1(ON))
41     {
42         for (i=1;i<5;i++)
43         {
44             ledFlash(ON,i);
45             wait(200);
46         }
47         for (i=0;i<5;i++)
48         {
49             ledFlash(OFF,i);
50         }
51         wait(200);
52     }
53     else
54     {
55         for (i=1;i<5;i++)
56         {
57             ledFlash(ON,i);
58             wait(200);
59             ledFlash(OFF,i);
60         }
61     }
62 } /*-----書き換
```

(*70)

ファイル一覧の表示方法

プロジェクト内のファイル一覧が表示されていない場合、プロジェクト名左側の「+」をクリックすると表示されます。



(*71)

波カッコ(ブレース)について

「{」と「}」は必ず1対1で対応します。(一つのプログラムの中に含まれる「{」の数と「}」の数は必ず一致)

対応する「{」と「}」を縦にそろえ、その中に記述するものは1段インデント(字下げ)すると見やすいです。

LED制御関数

ledFlash(command, ch);

①明滅モード
ON:点灯、OFF:消灯、TOGGLE:現在と反転

②LED番号
LED1, LED2, LED3, LED4
(1, 2, 3, 4でも指定可)

◆指定したLED(1~4)の明滅モード(ON/OFF/TOGGLE)を制御(*72)

例) ledFlash(ON, LED1); →LED1をONにする(点灯)
ledFlash(OFF, 3); →LED3をOFFにする(消灯)
ledFlash(TOGGLE, LED4); →LED4を反転させる
(ONならOFF, OFFならONに)

スイッチ(sw1)制御関数

sw1(ONまたはOFF);

◆SW1の状態(ON/OFF)を返す(*73)

例) if(sw1(ON)){
(何かの処理)
} →スイッチがONのとき, {}内の処理を実行

時間待ち関数

wait(time);

◆time(ミリ秒)の間, その行で処理を停止(*74)

例) wait(100);
→100ミリ秒=0.1秒の時間待ち
(100ミリ秒, 経過するまで処理が次の行へ進まない)

繰り返し処理

for(式A; 条件式B; 式C)

- ◆繰り返しをはじめるまえに式Aを実行
- ◆条件式Bを満たす間繰り返す
- ◆1周繰り返すごとに式Cを実行

例) for(i=0; i<3; i++)
{
(iの値)
}



繰り返しを始める前にi=0として、
iが3より小さい間繰り返し、
1周繰り返すごとにiを1ずつ増やす

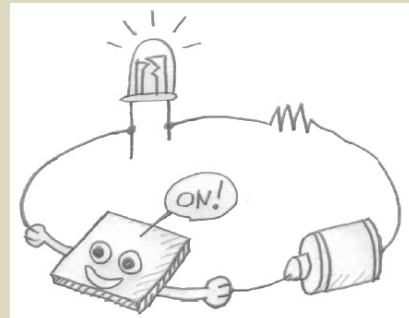
	1周目	2周目	3周目	4周目
例) for(i=0; i<3; i++) {	i=0実行 (i=0)	i=i+1実行 (i=1)	i=i+1実行 (i=2)	i=i+1実行 (i=3)
(iの値) }	i=0	i=1	i=2	実行されない (i<3が 満たされない)

⇒ {}の中のiは、i=0, i=1, i=2と変化し、結果として3周繰り返す。

(*72)

LED制御におけるマイコンの役割

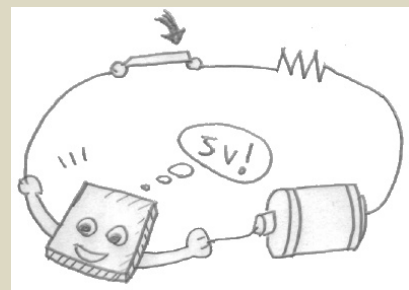
LED制御はデジタル出力の機能を使っています。



(*73)

スイッチ制御におけるマイコンの役割

スイッチの制御はデジタル入力の機能を使っています。



(*74)

時間待ち関数を入れる理由

どうして待つ必要があるのか?
待っていないと、ある処理から次の処理へと一瞬で移ってしまいます。例えばLEDを点灯させた後にwaitで少し時間を待たないと、人間の目では分からないくらい一瞬しか点灯しません。

スイッチ1がONのとき

LEDは1から順に点灯し、4つ点灯後、1から順に消灯

```
if (sw1 (ON))
```

```
{
  for (i=1; i<5; i++)
  {
    ledFlash (ON, i);
    wait (200);
  }
  for (i=0; i<5; i++)
  {
    ledFlash (OFF, i);
  }
  wait (200);
}
```

● ● ● ● ledFlash(ON,1); LED1点灯
wait(200); 0.2秒待つ

● ● ● ● ledFlash(ON,2); LED2点灯
wait(200); 0.2秒待つ

● ● ● ● ledFlash(ON,3); LED3点灯
wait(200); 0.2秒待つ

● ● ● ● ledFlash(ON,4); LED4点灯
wait(200); 0.2秒待つ

● ● ● ● ledFlash(OFF,1); LED1消灯
● ● ● ● ledFlash(OFF,2); LED2消灯
● ● ● ● ledFlash(OFF,3); LED3消灯
● ● ● ● ledFlash(OFF,4); LED4消灯
wait(200); 0.2秒待つ

0.2秒おきに
LEDが一つずつ
順番に点灯し、

こちらにはwaitが
入っていないので、
4つがほぼ同時に
消えます。

無限
ループ

スイッチ1がOFFのとき

LED1が点灯→消灯、その後LED2が点灯→消灯、LED3が...

```
else
```

```
{
  for (i=1; i<5; i++)
  {
    ledFlash (ON, i);
    wait (200);
    ledFlash (OFF, i);
  }
}
```

● ● ● ● ledFlash(ON,1); LED1点灯
wait(200); 0.2秒待つ

● ● ● ● ledFlash(OFF,1); LED1消灯

● ● ● ● ledFlash(ON,2); LED2点灯
wait(200); 0.2秒待つ

● ● ● ● ledFlash(OFF,2); LED2消灯

● ● ● ● ledFlash(ON,3); LED3点灯
wait(200); 0.2秒待つ

● ● ● ● ledFlash(OFF,3); LED3消灯

● ● ● ● ledFlash(ON,4); LED4点灯
wait(200); 0.2秒待つ

● ● ● ● ledFlash(OFF,4); LED4消灯

Waitが入っていない
ので、LED1を消した
直後にLED2をすぐ
点灯させています。

無限
ループ

5-3 LED_Switchチャレンジ

- (1) 二倍速で光が動くプログラムを作ってみよう
- (2) 逆に右から左に光が動くプログラムを作ってみよう
- (3) 自分のオリジナルの点滅パターンを作成してみよう

やっぱりロボットの目は光らなくちゃネ！

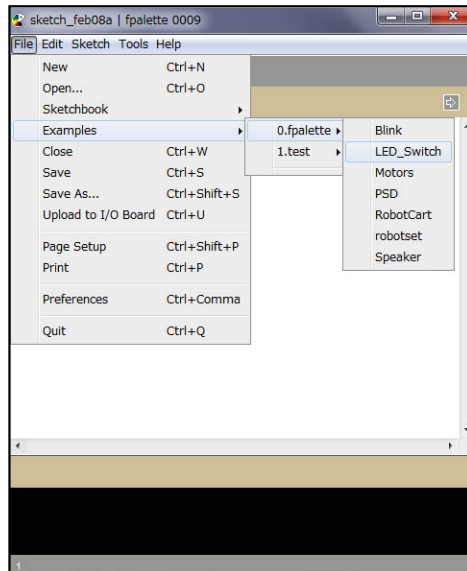


5-4 f-palette IDE プログラムの実行

次に、「f-palette」で、同じようにLEDを点滅させてみましょう。

まずは、4-4節のいちばん最後の状態、すなわちf-palette IDEにて開発ができる状態に戻します。

「f-palette.exe」は立ち上がっていますね。では、「File」→「Example」→「0.fpalette」→「LED_Switch」を開きます。



そうしたら、サンプルプログラムを実行した時と同様に、「Sketch」→「Verify/Compile」でビルド確認します。「Done compiling」と表示されていますね。特に問題はないはずです。

そして、「File」→「Upload to I/O Board」でマイコンに書き込みを行います。「Binary sketch size: **** bytes (of a 32256 byte maximum)」と表示されたタイミングで、CPUボードのリセットボタンを押下してください。この作業は、今後も、毎回プログラムを書き込むたびに行います。



← 注目！

どうでしょうか。CCSで動かした時のように、LEDは点滅していますか？ SW1を切り替えて、点滅の仕方の変化も確認してみましょう。

さてそれでは、CCS開発環境でのプログラムと、f-palette IDEでのプログラムを見比べてみましょう。左がCCS、右がf-palette IDEです。

```

14 #include "DSP28x_Project.h"
15 #include "NolibM.h"
16 #include <stdio.h>
17
18 volatile struct EPWM_REGS *ePWM[] =
19 {
20     &EPwm1Regs,
21     &EPwm2Regs,
22     &EPwm3Regs,
23     &EPwm4Regs,
24     &EPwm5Regs,
25     &EPwm6Regs,
26     &EPwm7Regs,
27     #if (DSP2804x_DEVICE_H)
28     &EPwm8Regs
29     #endif
30 };
31
32 void main(void)
33 {
34     int i;           // for文用 変数宣言
35     initNolibM();    // 初期化
36     while(1)         // 無限ループ
37     {
38         if(sw1(ON))
39         {
40             for(i=1;i<5;i++)
41             {
42                 ledFlash(ON,i);
43                 wait(200);
44             }
45             for(i=0;i<5;i++)
46             {
47                 ledFlash(OFF,i);
48                 wait(200);
49             }
50         }
51         else
52         {
53             for(i=1;i<5;i++)
54             {
55                 ledFlash(ON,i);
56                 wait(200);
57                 ledFlash(OFF,i);
58             }
59         }
60     }
61 }
62
63

```

CCSプログラム

```

#include "fpalette_pin.h"
#include "palette.h"

int i;

void setup()
{
    palette_init();
}

void loop()
{
    if(sw1(ON))
    {
        for(i=1;i<5;i++)
        {
            ledFlash(ON,i);
            wait(200);
        }
        for(i=0;i<5;i++)
        {
            ledFlash(OFF,i);
            wait(200);
        }
    }
    else
    {
        for(i=1;i<5;i++)
        {
            ledFlash(ON,i);
            wait(200);
            ledFlash(OFF,i);
        }
    }
}

```

Arduinoプログラム

条件分岐

スイッチ制御関数

繰り返し処理

LED制御関数

時間待ち関数

実行プログラムのメイン関数部分を比べると、ほとんど変わらないことが分かりますね。両者の違いは、CCSでは"while"で既述されているのに対し、f-palette IDEは"loop"であるところです。どちらもボードの電源が切られるまで、何度も繰り返し実行される(無限ループ)ことになります。

ですから当面は、インタフェースの使い方に慣れている方で開発してみることをお勧めします。

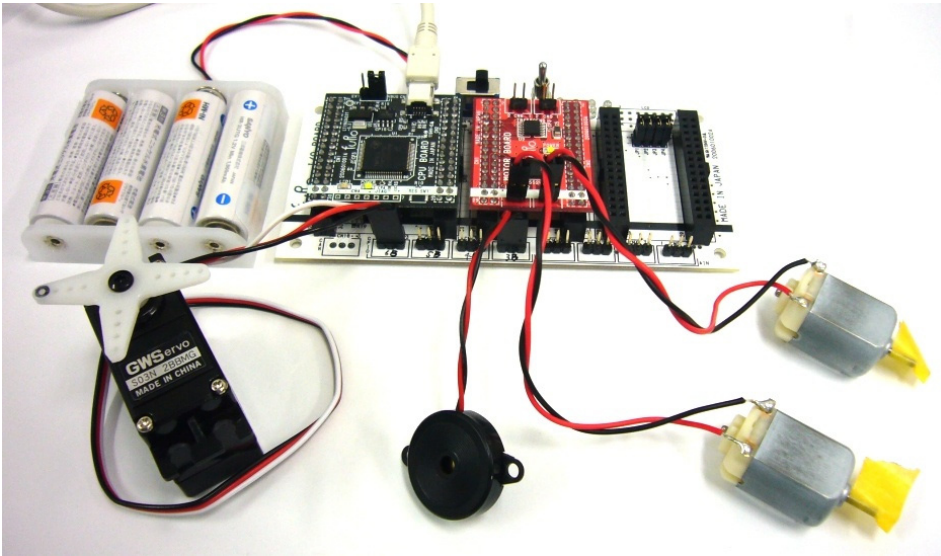
7章 モータを回してみよう –PWM&モータドライバでモータを駆動する

概要

ロボットの『動く』部分を担当するのがモータです。モータは皆さんも一度は目にしたことがあるでしょう。電池につなげば軸がぐるぐる回る部品ですね。

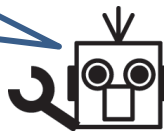
モータを思い通りに回せるようになると、車輪を回したり、ロボットの関節部分に組み込んで腕や脚を動かしたりと、いろんなことができるようになります。モータを思い通りに回すために必要なのが、前回使ったPWM機能と、本章で作成するモータドライバに入っているHブリッジ回路です。

この章ではモータの仕組みを知り、PWM機能とモータドライバ(Hブリッジ)を使ってモータを回してみましょう。



- 7-1 モータ&モータドライバの組み立て
- 7-2 サンプルプログラム『Motors』
- 7-3 プログラムの中身を見てみよう
- 7-4 Motorチャレンジ
- 7-5 f-palette IDEのプログラムの実行
- 7-6 モータをコントロールする仕組み
- 付録 モータの構造と回る理由

ロボットの車輪だったり、
腕とか足を自由に動かせるゾ！

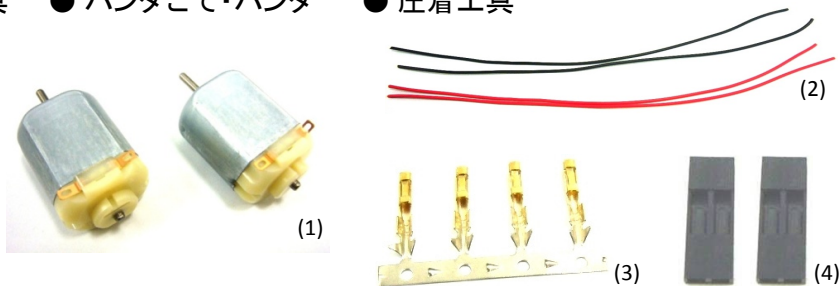


7-1 モータ&モータドライバの組み立て

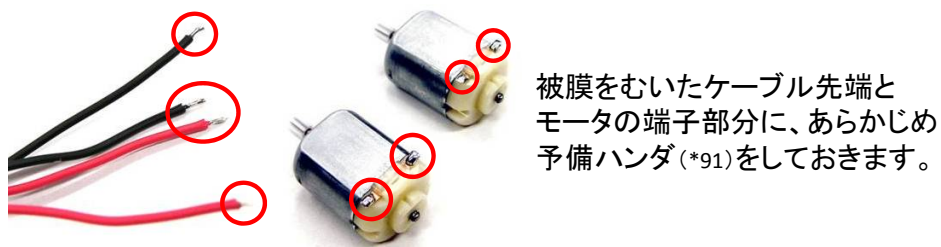
① モータの組み立て

用意するもの※

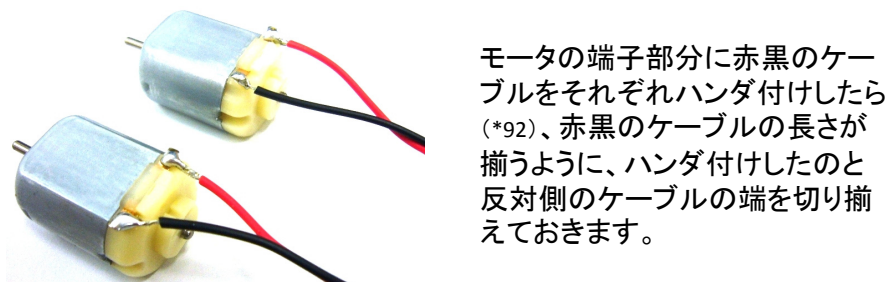
- | | | |
|----|-------------------|-----------------------|
| 部品 | (1) DCモータ(*90) 2個 | (2) ケーブル赤・黒 各2本 |
| | (3) コネクタピン 4個 | (4) コネクタハウジング(2ピン) 2個 |
| | (5) 電池ボックス 1個 | |
| 道具 | ● ハンダごて・ハンダ | ● 圧着工具 |



手順1 モータにケーブルをハンダ付けする。

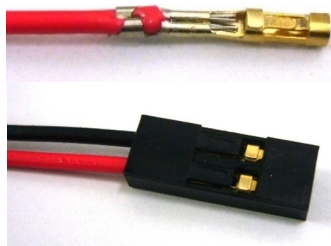


被膜をむいたケーブル先端とモータの端子部分に、あらかじめ予備ハンダ(*91)をしておきます。



モータの端子部分に赤黒のケーブルをそれぞれハンダ付けしたら(*92)、赤黒のケーブルの長さが揃うように、ハンダ付けしたのと反対側のケーブルの端を切り揃えておきます。

手順2 ケーブルにコネクタピンを圧着し、ハウジングに差し込む。



圧電スピーカの組み立ての手順2～4と同様に、切り揃えたケーブル端の被膜をはがし、コネクタピンを圧着します。

圧着したらハウジングに差し込みます。

手順3 完成！

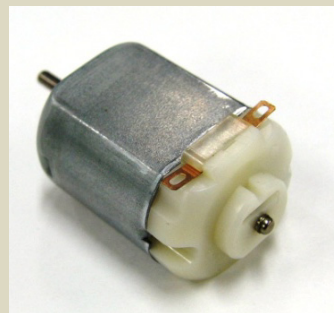


※部品は後述12章、秋月電子、千石電商にてすべて購入することができます。

千石電商

- (1) Mabuchi FA-130RA相当F-13-10249-5等
 (2) φ1.2mmケーブル
 (3) 2550シリーズ信号伝達コネクタ用ピン
 (4) 2550シリーズ信号伝達コネクタ(黒) 1x2
 (5) 電池ボックス(単3X4・リード線付) M C304-3リード線付
 圧着工具 ホーザンP-706

(*90)
DCモータとは



直流電流(Direct Current)で駆動されるモータ。内部にブラシという機械部品があるもの(ブラシ付きDCモータ)と、機械部品の代わりに電子部品でブラシの機能を置き換えたもの(ブラシレスDCモータ)があります。ここではブラシ付きDCモータをDCモータと呼びます。

(*91)
《コツ》予備ハンダ

ケーブルと端子、ケーブルとランドなどをハンダ付けする際に、それら二つをくっつけてからハンダ付けするのではなく、あらかじめケーブル先端や端子それぞれにハンダを付けておくことを言います。予備ハンダをしておくと、ケーブルと端子をくっつける際に作業しやすくなります。

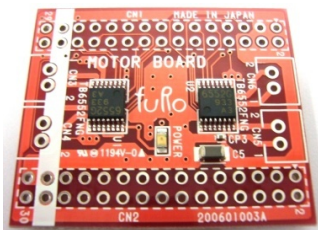
(*92)
赤・黒ケーブルの順番

どちらの端子にどちらの色のケーブルをハンダ付けするのか、あるいはハウジングに差し込む際の赤・黒の順番はどちらが正しいのか。これはどちらでも大丈夫です。ただし、正回転と逆回転の向きが反対になりますので、2つのモータ間では揃えておいた方が分かりやすいです。

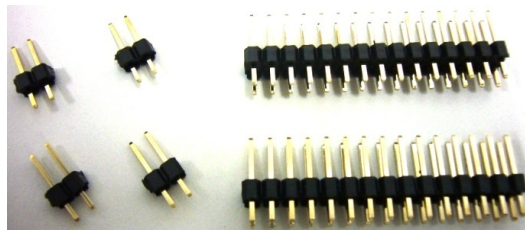
② モータドライバ組み立て

用意するもの

- 部品 (1) モータドライバボード 1枚
(2) ピンヘッダ 2x1ピン 4個、2x15ピン 2個
- 道具 ● ハンダごて・ハンダ



(1)

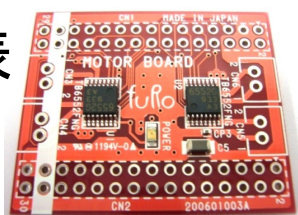


(2)

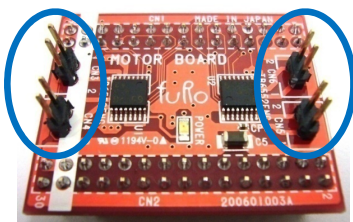
a. モータドライバボードの組み立て

手順1 モータドライバボード・表面にピンヘッダをハンダ付けします。

表



『fuRo』の文字が書いてある方が表面です。



2x1ピンをCN3,CN4,CN5,CN6の4カ所へ、長い方が表面に出るように差し込み、裏側からハンダ付けします。

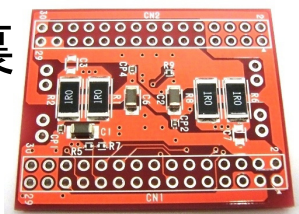
(*93)

《コツ》長いピンヘッダのハンダ付け

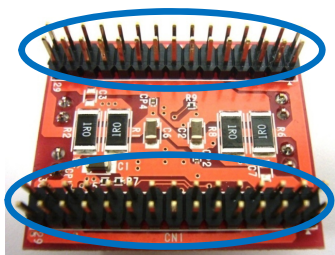
長いピンヘッダをハンダ付けするときは、両端の部分をまずハンダ付けして、ピンヘッダが曲がって取り付けられていないか確認してから、全体のハンダ付けをしましょう。曲がってハンダ付けしてしまうと、モータドライバボードがI/Oボードに差さらない恐れがあります。

手順2 モータドライバボード・裏面にピンヘッダをハンダ付けします。(*93)

裏

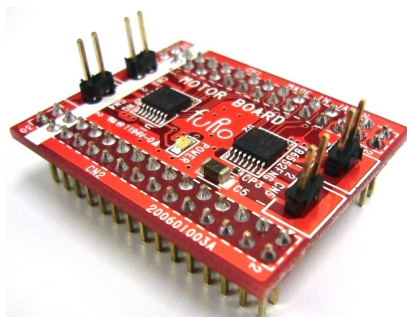


太い白ラインが印刷されていない、黒い部品が4つ並んでいるほうが裏面です。



2x15ピンを、CN1,CN2の2カ所へ、長い方が裏面に出るように差し込み、短いほうを表側からハンダ付けします。

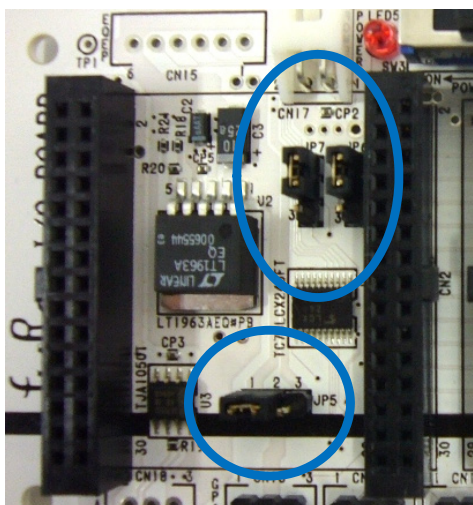
手順3 完成！



モータ・モータドライバが完成したら、f-palette I/Oボードに接続します。

b. モータ・モータドライバボードの接続

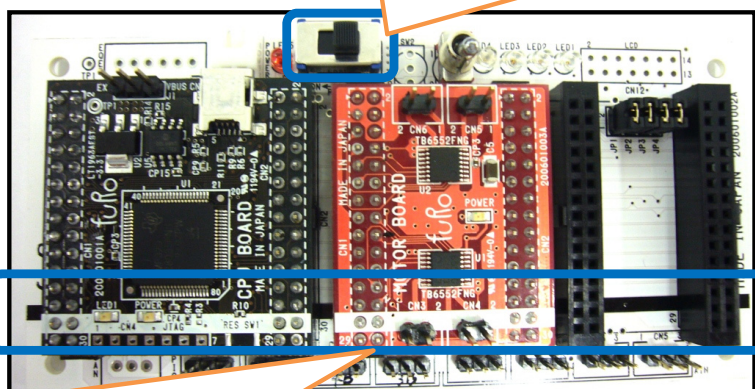
手順1 JP(ジャンパピン)を確認します。



マイコンボードの下にあるJP5、JP6、JP7のジャンパピンは、すべて1-2をショート(接続)させます。

手順2 マイコン・モータドライバボードをI/Oボードに接続します。

スイッチSW3をOFF側にしておきます

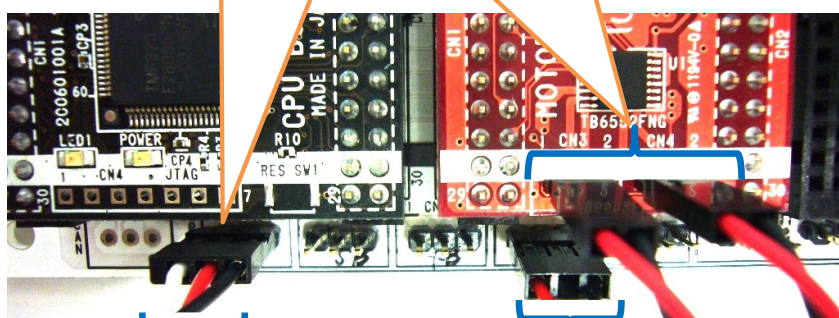


I/Oボードの黒いラインと、マイコンボード・モータドライバボードの白いラインが、必ず一直線に揃うように差し込んでください！！

手順3 DCモータ、サーボモータ(*94)と圧電スピーカーを接続します。

D4 : サーボモータ

CN3, CN4 : DCモータ



白 赤 黒

赤 黒

D1: 圧電スピーカ

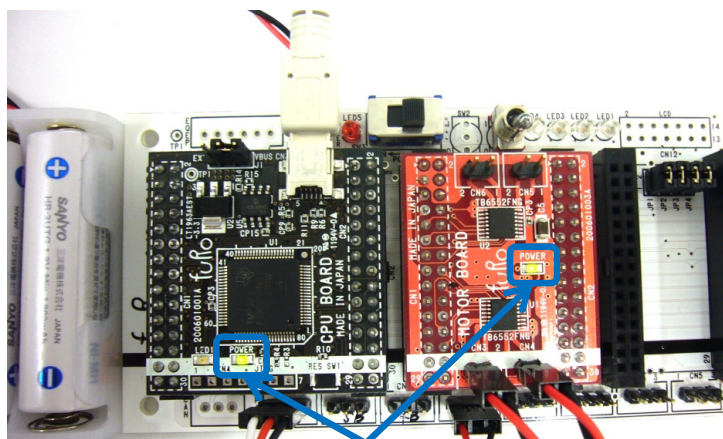
(*94)
サーボモータとは



DCモータに、回転角度を測るセンサや、歯車等で回転速度を抑える減速機、制御回路等を組み合わせて一体化したモータ。ただ回転するだけではなく、与えられた目標角度まで回転し、その角度を保持することができます。

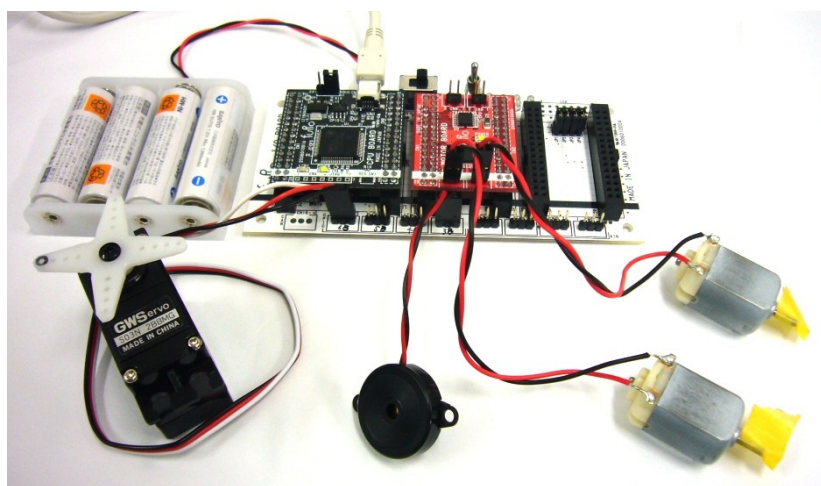
角度センサで得られる現在の角度情報を基に、目標の角度になるようにモータの動きを制御することを「フィードバック制御」と言います。

手順4 電池ボックスのケーブルとUSBケーブルを接続します。



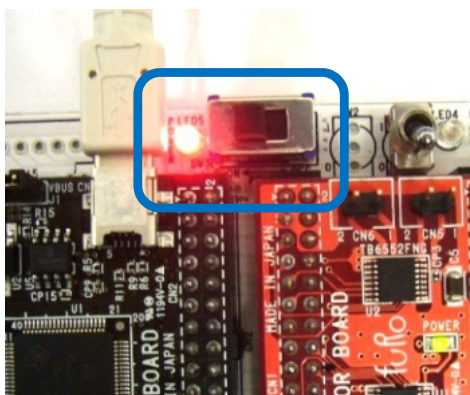
マイコン・モータドライバボードのLEDが点灯します

手順5 完成！



7-2 サンプルプログラム『Motors』の実行

5-1と同様の手順でサンプルプログラム『Motors』をインポートして、Build→Load→Runを行ってみましょう。



I/Oボードの電源(SW3)をONにすると、モータが回り出すはずです。

モータを止める場合は、まずI/Oボードの電源(SW3)をOFFにしてから、CCS上の実行停止ボタンを押しましょう。

(*95)

トラブルシューティング

- 全部のモータが回らない
→I/Oボードの電源(SW3)を入れ忘れていませんか？
→電池は十分にありますか？
- サーボモータが回らない
→サーボモータはD4にきちんと刺さっていますか？
→刺す向きは間違っていないですか？
- DCモータが回らない
→モータドライバボードの刺す向きは合っていますか？
→DCモータの刺す場所は？
→モータドライバボードを取り替えてみたらどうですか？
- DCモータが激しく回る
→手順を間違えていませんか？

Check!! (*95)

- モータは回っている？
- 圧電スピーカーから音は出ている？
- SW1を切り替えたら、モータの回り方が変化する？

7-3 プログラムの中身を見てみよう

サンプルプログラム『Motors』を実行して、DCモータとサーボモータを回転させることができました。それをプログラムでどうやって実現しているのか「Motors.c」を見てみましょう。

40行目から65行目までは
for(;;)による無限ループ(*96)

```
39 for(;;) // 無限ループ
40 { /*-----ここ
41     if (sw1 (ON))
42     {
43         motor (+30, MotCN3);
44         motor (+60, MotCN4);
45
46         servo (+90, D4);
47         wait (1000);
48
49         servo (0, D4);
50         wait (1000);
51
52         mario ();
53     }
54     else
55     {
56         motor (-30, MotCN3);
57         motor (-20, MotCN4);
58
59         servo (-90, D4);
60         wait (1000);
61
62         servo (0, D4);
63         wait (1000);
64     }
65 }
```

if(sw1(ON))
SW1が「ON」ならば
{ }の中(42~53行目)を実行

else
上記の場合でなければ
(SW1が「OFF」ならば)
{ }の中(55~64行目)を実行

(*96)
繰り返し処理 for による無限ループ

for(;;)は、5章で説明した

for(式A; 条件式B; 式C)

の括弧内のすべての式が無いという意味です。繰り返しをやめる条件も、繰り返しの前・最中に実行される式もないため、無限に繰り返しが行われる＝無限ループとなります。ここまで使ってきた while(1) を、違う書き方で実現しています。

DCモータ駆動関数

motor(speed, ch);

①回転速度

②モータ番号

-100~+100の範囲で指定
(最大電圧を加えたときの速度を100とする)

MotCN3, MotCN4

正の整数: 正の方向(CW)に回転(正回転)(*97)

負の整数: 負の方向(CCW)に回転(逆回転)

0 : ブレーキ, 回転の停止

◆モータドライバボードの指定したコネクタ(CN3, CN4)に接続したDCモータを駆動させる。

例) motor(30, MotCN3);

→CN3に接続されたモータを、30の速度で正転で駆動させる。

motor(-80, MotCN4);

→CN4に接続されたモータを、80の速度で逆転で駆動させる。

motor(0, MotCN3);

→CN3に接続されたモータの回転を停止させる。

(*97)
モータの回転方向

正転と逆転が反対になってる人はいませんか？

モータコネクタをモータに差し込む向きによって、モータに流れる電流の方向が変わるため、モータの回転方向も変わります。プログラムする際に、実際のモータの回転方向に合わせて調整してください。

サーボモータ駆動関数

`servo(angle, ch);`

①回転角度
-100～+100(deg)の範囲で指定

②出力コネクタ
D1, D2, D3, D4

◆指定コネクタ(D1, D2, D3, D4)に接続したサーボモータを駆動する。

例) `servo(+90,D4);`

→D4のサーボモータを90度の位置まで回転させる

`servo(-45,D4);`

→D4のサーボモータを-45度の位置まで回転させる

`servo(0,D4);`

→D4のサーボモータを0度(初期位置)まで回転させる

スイッチ1がONのとき

```
41  if (sw1 (ON) )
42  {
43      motor (+30, MotCN3);
44      motor (+60, MotCN4);
45
46      servo (+90, D4);
47      wait (1000);
48
49      servo (0, D4);
50      wait (1000);
51
52      mario();
53  }
```

CN3に接続されたDCモータを
正転方向に30の速度で駆動

CN4に接続されたDCモータを
正転方向に60の速度で駆動

D4に接続されたサーボモータを
+90度まで回転させる

1000[ms] = 1秒 待つ
その間DCモータは回転を続け、
サーボモータは目標角度まで回転する

D4に接続されたサーボモータを
0度(初期位置)まで回転させる

1000[ms] = 1秒 待つ

スピーカから音楽を流す

7-4 Motorチャレンジ

(1) DCモータの回転速度(`motor(speed,ch)`の`speed`の値)や、サーボモータの回転角度(`servo(angle,ch)`の`angle`の値)を変えてみよう！

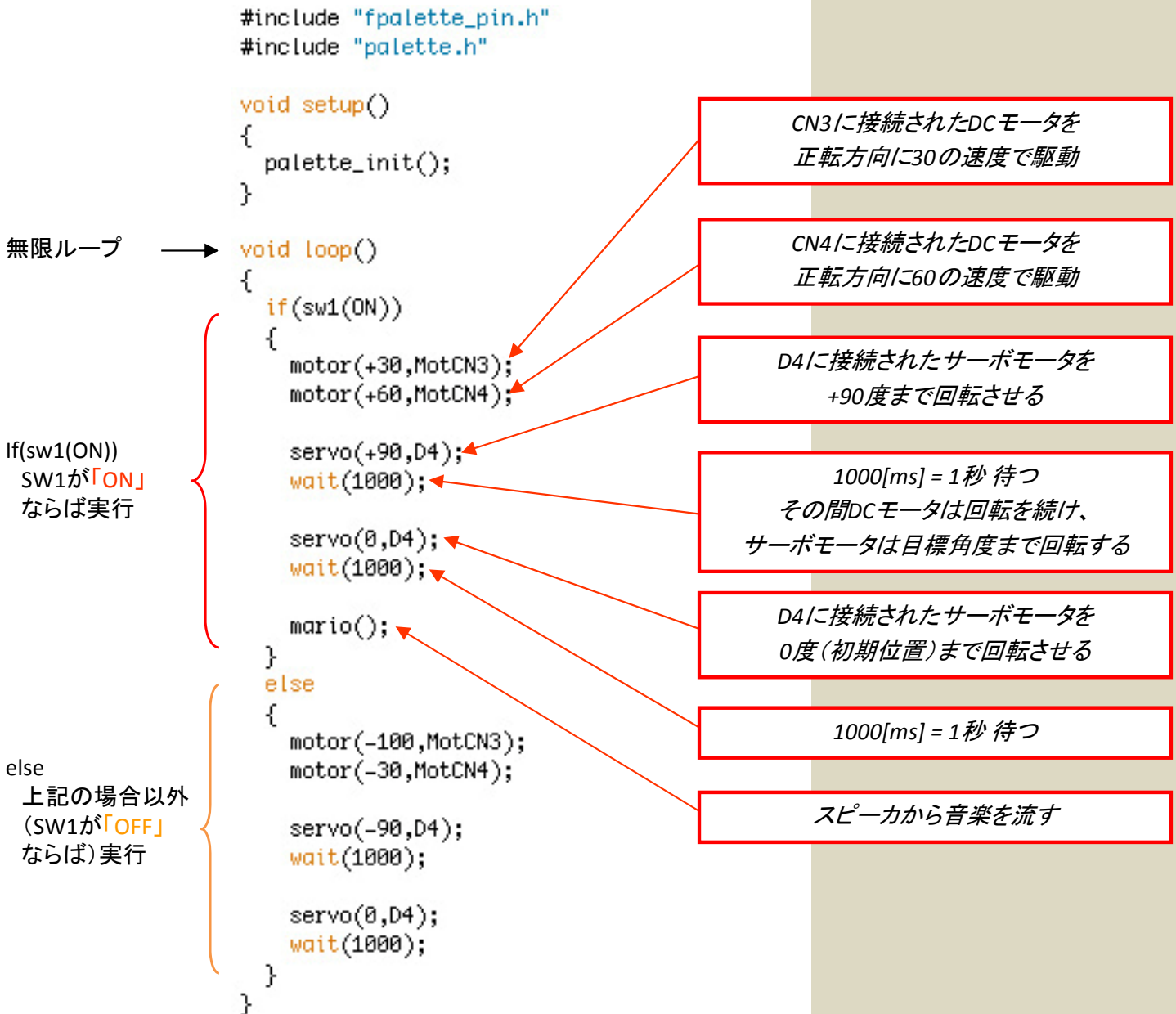
(2) 自分なりにモータを回転させてみよう！

7-5 f-palette IDEのプログラムの実行

5-4と同じ手順で、f-palette IDEの開発環境下、サンプルプログラム『Motors』を開き、Verify/Compileした後、書き込みを行ってみましょう。I/Oボードの電源(SW3)をONにすると、モータが回り出すはずです。

モータの動きや圧電スピーカーから出る音をチェックしましょう。また、SW1を切り替えて、モータの回り方の変化を確かめましょう。

以下、サンプルプログラムの中身です。DCモータ駆動関数やサーボモータ駆動関数の使い方は、CCSと変わりません。

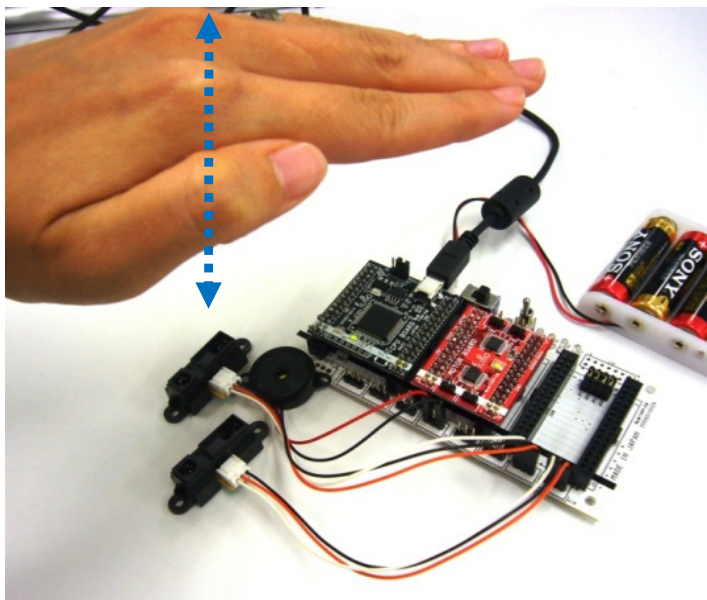


さあ、プログラムが理解できたら、じゃんじゃん好きなように手を加えていってみよう。

8章 センサでいろいろ測ってみよう ―A/D(アナログ・デジタル)変換

概要

ロボットの『感じる』部分を担当するセンサのひとつであるPSD(赤外線測距)センサ(*104)を使ってみます。センサを使うためには、センサが出力する連続的な電圧(アナログ電圧)を、マイコンが扱いやすいデジタル値に変換する、「A/D(アナログ・デジタル)変換」という機能が必要となります。この章ではA/D変換の仕組みを知り、PSDセンサを組み立て、f-palette でセンサの値を読み取ってみましょう。



- 8-1 A/D変換とは？
- 8-2 PSDセンサ組み立て
- 8-3 サンプルプログラム『PSD』
- 8-4 プログラムの中身を見てみよう
- 8-5 センサの出力データを確認してみよう
- 8-6 PSDチャレンジ
- 8-7 f-palette IDEのプログラムの実行

8-1 A/D(アナログ・デジタル)変換とは？

A/D変換の『アナログ』『デジタル』という言葉、皆さんもあちこちで耳にしたことがあると思います。アナログ(Analog)というのは重さや長さなど、厳密に測ればいくらでも細かく測れる、連続した値のことを言います。それに対してデジタル(Digital)は、目盛りが決まっていいて数えることができる、とびとび(離散的)の値のことを言います。

例えば、アナログ表示の体重計だと、針の位置は目盛りと目盛りの間を連続的に変化するため、例えば針が345kgと346kgのメモリの間に会ったら、細かく見ればいくらでも細かく見ることができます。でも、見る人によっては345kgとも、346kgとも言われることもある、曖昧な表示です。それに対してデジタル表示の体重計では、誰が見ても同じ345.0kgと表示されます。本当は344.9kgだったり、345.1kgかもしれませんが、それを345.0kg!と自分の目盛りで決めてしまうのがデジタルです。

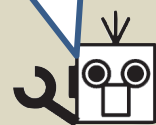
(*104)

PSD(赤外線測距)センサとは

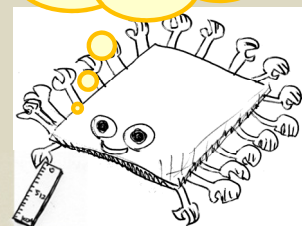


対象物までの距離に応じたアナログ電圧を出力するセンサ。片側のレンズ内部から赤外線を出し、障害物に当たって跳ね返ってきた赤外線を、反対側のレンズの内部にある光位置検出素子(PSD: Position Sensitive Detector)で検出します。検出した光のスポット位置から、障害物までの距離が三角測量の原理によって計算できます。

A/D変換?????
気にスルナ!
8-2へ行ってしまエ!



「0」「1」で
表現したい!



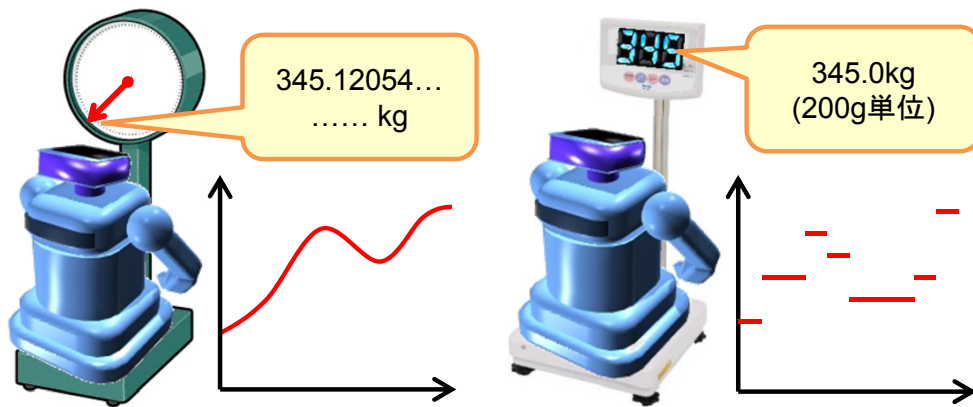


図8-1 アナログとデジタル

PSDセンサを始め、f-palette ボードで利用できるセンサの多くは、測定した結果を連続的なアナログ電圧として出力します。しかしながらマイコンは、白黒がはっきりしているスイッチのON/OFFなら得意なのですが、割り切れない連続した値を扱うのは苦手…。そこでセンサから入ってきたアナログの値を、マイコンが持っている定規の目盛りを使って、マイコンが扱いやすいデジタルの値にするのがA/D変換です。

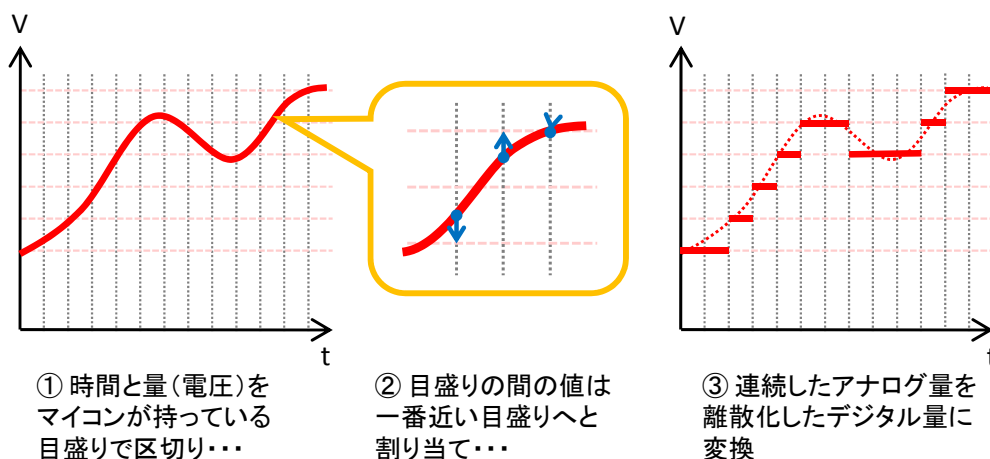


図8-2 A/D変換の流れ

マイコンが持っている定規の目盛りの数は、マイコンの種類によってさまざまです。f-paletteマイコンボードに搭載されているPiccoloの目盛りの数は12[bit] = 4096、つまりセンサから出力される0~5[V]の電圧は、0~4095のデジタル値に変換されます。例えば、センサから最大の5.0[V]が出力された場合、A/D変換されてマイコンに取得されるデータは4095に、3.0[V]が出力された場合は2457になります。

つまり、この目盛りの数が多いほど、同じセンサを使用した場合にマイコンが取得できるデータが細くなり、センサが出力した結果をより正確にマイコンで用いることができます。(*105)

それでは実際に、距離を測るPSDセンサを組み立て、マイコンのA/D変換機能を使ってセンサの出力を確認してみましょう。

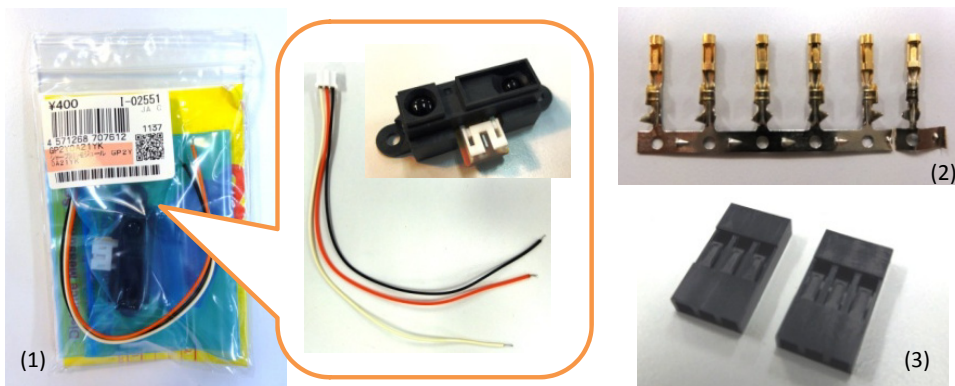
(*105) 目盛りの数と精度

この目盛りの数(Piccoloだと12[bit])のことを、A/D変換の「分解能」といいます。どれだけ細かく値を見ることができるかどうかを表現するものです。

8-2 PSDセンサ組み立て

用意するもの※

- 部品 (1) PSDセンサ(ケーブル付き) 2個(*106) (2) コネクタピン 6個
 (3) コネクタハウジング(3ピン用) 2個
 道具 ● 圧着工具(ホーザンP-706等)



手順1 ケーブルにコネクタピンを圧着し、ハウジングに差し込む。

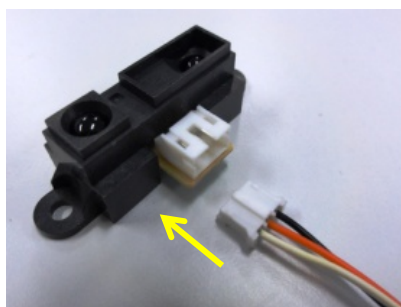


圧電スピーカの組み立て手順2～4と同様に、ケーブル端の被膜をはがし、コネクタピンを圧着します。

圧着したら差し込む順番に注意してハウジングに差し込みます。

コネクタピンの金属部分がしっかり入るように、カチッと音が鳴るまで奥へ差し込みます。

手順2 PSDセンサ側のコネクタにケーブルを接続する。



ケーブル側のコネクタの凸部分とセンサ側コネクタに入っている溝が合うように差し込みます。

手順3 完成！



※部品は後述12章、秋月電子、千石電商にてすべて購入することができます。

秋月電子

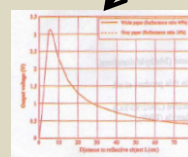
- (1) シャープ測距モジュール GP2Y0A2 1YK
 千石電商
 (2) 2550シリーズ信号伝達コネクタ用ピン
 (3) 2550シリーズ信号伝達コネクタ(黒) 1x3

(*106)

PSDセンサの空き袋

英語だからと捨てないで・・・

測距センサと一緒に入っていたシートは情報の宝庫！

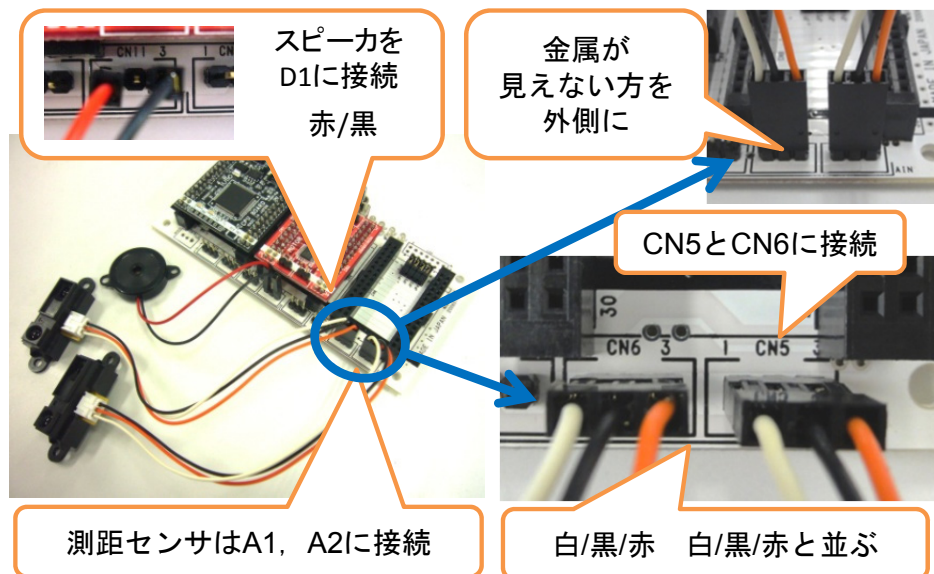


距離に応じてどんな電圧が出てくるか？



センサのピンに何を入力するか？

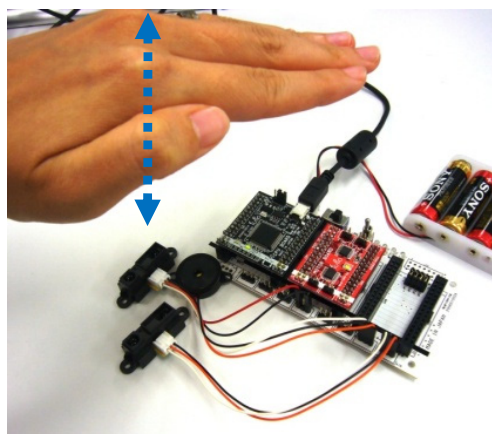
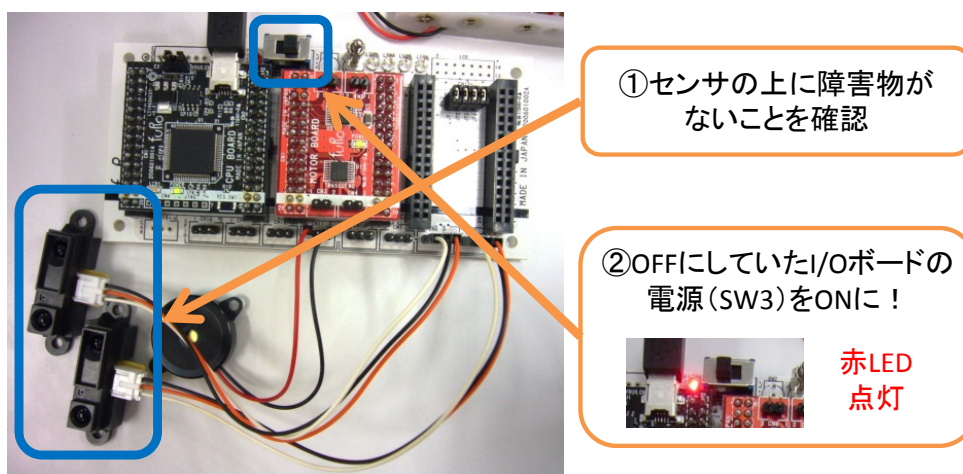
PSDセンサが2個完成したら、f-palette I/Oボードに接続します。



前章の接続手順1～4と同様に、ジャンパピンを確認してからマイコンボードを接続し、電池とUSBを接続して電源を入れます。

8-3 サンプルプログラム『PSD』の実行

5-1と同様の手順でサンプルプログラム『PSD』をインポートしてBuild→Load→Runを行ってみましょう。



Check!!

スピーカーからどんな音が出る？
I/OボードのLEDはどう光る？
距離を変えるとどうなる？

実行を止める場合には、まずI/Oボードの電源(SW3)をOFFにしてから、CCS上の実行停止ボタンを押しましょう。

8-4 プログラムの中身を見てみよう

サンプルプログラム『PSD』を実行して、PSDセンサと障害物の距離を検出し、その距離に応じてLEDを点滅させ、スピーカから音を出すことができました。それをプログラムでどうやって実現しているのか「PSD.c」を見てみましょう。

条件分岐

if(条件1){(処理1)}

◆条件1が満たされたときにだけ、(処理1)を実行する

else if(条件2){(処理2)}

◆ifとセットで用いられ、条件1が満たされず、かつ条件2が満たされたときにだけ、(処理2)を実行する

else{(処理3)}

◆ifやelse ifとセットで用いられ、ifやelse ifの条件が満たされなかったときに(処理3)を実行する

センサデータ取得関数

(変数) = sensor(ch);

①取得データ

②センサが接続されたコネクタを指定
A1, A2, A3, A4

センサの出力電圧をA/D変換した値(0~4095)が代入される(*107)

◆指定したコネクタ(CN5,CN6)に接続されたセンサの出力電圧をA/D変換した値(0~4095)を、変数に代入する。

例) sen1 = sensor(A1);

→A1に接続されたセンサの出力をA/D変換して、変数sen1に代入

sen2 = sensor(A2);

→A2に接続されたセンサの出力をA/D変換して、変数sen2に代入

37行目から94行目までは
for(;;)による無限ループ

39, 40行目

sen1=sensor(A1);
sen2=sensor(A2);

でA1, A2のセンサのA/D変換値を取得

46行目から、A1のセンサの値sen1に応じて
if/else if/elseを使って実行する処理
を変える

(beep()を使って音を鳴らす)

69行目からも同様に、A2のセンサの値
sen2に応じて if/else if/elseを使って実
行する処理を変える

(ledFlash ()を使ってLED点滅)

```
36 for(;;) // 無限ループ
37 { /*-----ここから書き換える!!!-----
38
39     sen1 = sensor(A1); // A1 (CN5)に接続された
40     sen2 = sensor(A2); // A2 (CN6)に接続された
41
42     /*
43      * A1 (CN5)に接続されたセンサが反応した場合はス
44      */
45
46     if(sen1 > 2000){ // sen1の値が2000より大き
47         beep(SHI,D1,100);
48     }
49     else if(sen1 <= 2000 && sen1 >1800){ //
50         beep(RA,D1,100);
51     }
52     else if(sen1 <= 1800 && sen1 > 1600){ //
53         beep(SO,D1,100);
54     }
55     else if(sen1 <= 1600 && sen1 > 1400){ //
56         beep(FA,D1,100);
57     }
58     else if(sen1 <= 1400 && sen1 > 1200){ //
59         beep(MI,D1,100);
60     }
61     else if(sen1 <= 1200 && sen1 > 1000){ //
62         beep(RE,D1,100);
63     }
64     else if(sen1 <= 1000 && sen1 > 900){ //
65         beep(DO,D1,100);
66     }
67     else{// sen1の値が900以下の場合(センサから
```

(*107)

PDSセンサの測定値

今回用いているPSDセンサの測定可能距離は10~80[cm]で、出力電圧は距離10cmのときに3.5 [V] 程度、80cm以上離れたときにほぼ0[V]となります。A/D変換値にしておよそ2200~0 程度です。

論理積 “&&” と論理和 “||”

if([条件A] && [条件B]){(処理1)}

◆条件Aと条件Bが両方とも成り立つとき(論理積)だけ、(処理1)を実行する。

例) if(sen1 <= 2000 && sen1 > 1800){ (49行目)
(条件A) (条件B)

→sen1の値が2000以下(条件A)で、**かつ**1800より大きい(条件B)のとき、{}内の処理が実行される。

if([条件A] || [条件B]){(処理1)}

◆条件Aと条件Bのどちらかが成り立つとき(論理和)に、(処理1)を実行する。

例) if(sen1 > 2000 || sen2 > 2000){
(条件A) (条件B)

→sen1の値が2000より大きい(条件A)か、**もしくは**sen2の値が2000より大きい(条件B)のとき、{}内の処理が実行される。

A1に接続されたセンサの値sen1に応じたプログラムをより詳しく見てみると、

```
if(sen1 > 2000){// sen1の値が2000より大きい  
  beep(SHI,D1,100);  
}  
else if(sen1 <= 2000 && sen1 >1800){//  
  beep(RA,D1,100);  
}  
else if(sen1 <= 1800 && sen1 > 1600){/  
  beep(SO,D1,100);  
}  
else if(sen1 <= 1600 && sen1 > 1400){/  
  beep(FA,D1,100);  
}  
else if(sen1 <= 1400 && sen1 > 1200){/  
  beep(MI,D1,100);  
}  
else if(sen1 <= 1200 && sen1 > 1000){/  
  beep(RE,D1,100);  
}  
else if(sen1 <= 1000 && sen1 > 900){//  
  beep(DO,D1,100);  
}  
else{// sen1の値が900以下の場合(センサから)
```

sen1の値が2000より大きい場合

【シ】をCN11から100[ms]出す

sen1が1800より大きく2000以下

【ラ】をCN11から100[ms]出す

sen1が1600より大きく1800以下

【ソ】をCN11から100[ms]出す

sen1が1400より大きく1600以下

【ファ】をCN11から100[ms]出す

sen1が1200より大きく1400以下

【ミ】をCN11から100[ms]出す

sen1が1000より大きく1200以下

【レ】をCN11から100[ms]出す

sen1が900より大きく1000以下

【ド】をCN11から100[ms]出す

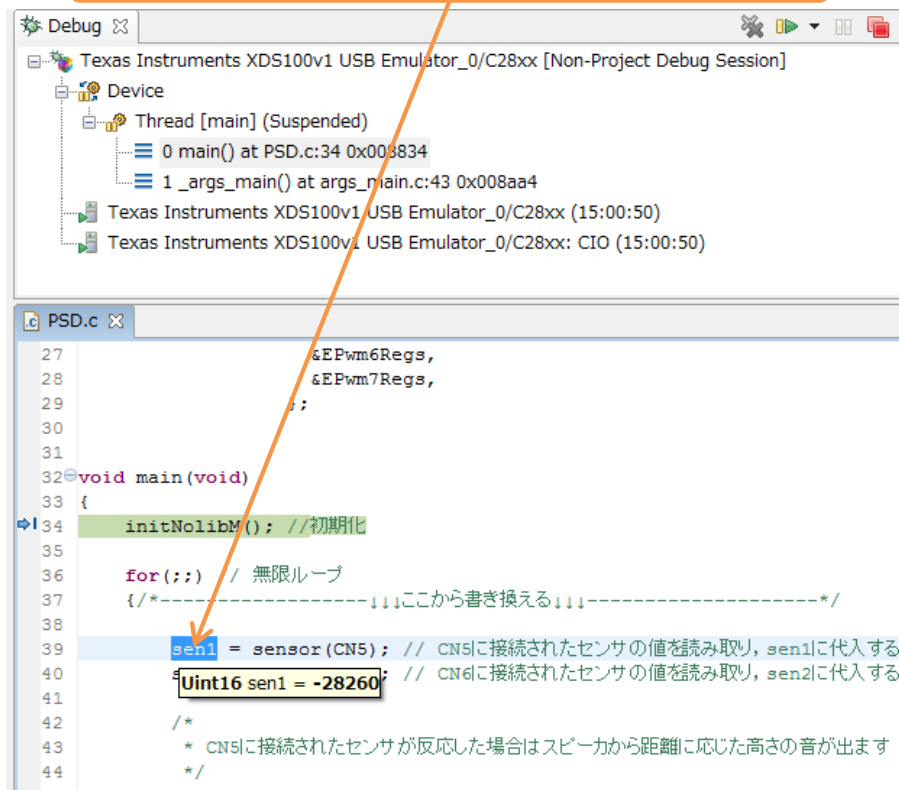
上の条件に当てはまらない(sen1の値が900以下)の場合
(何もしない)

A1に接続されたセンサに障害物が遠くから近づいてくるに従って、シ…
ラ…ソ…と、スピーカから出てくる音がだんだん、高い音から低い音へと変
化していきます。

8-5 センサの出力データを確認してみよう

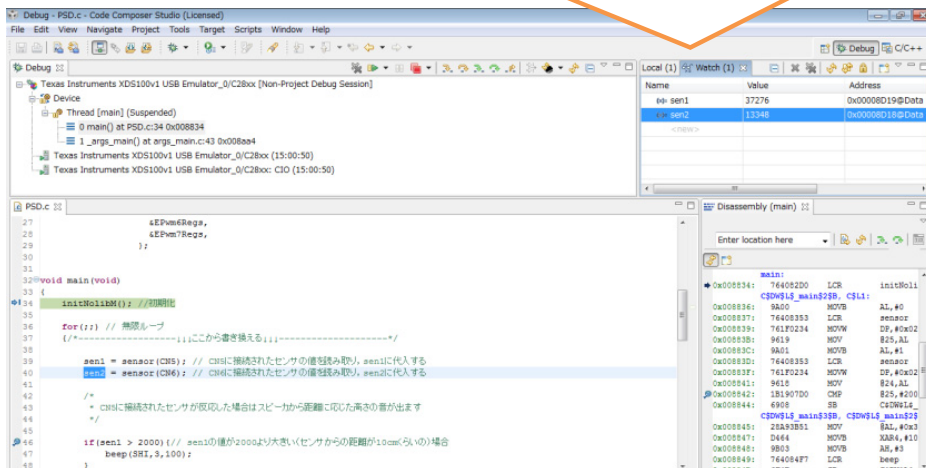
CCSには他の開発環境にはなかなかない機能として、プログラムを実行（Run）しながら、プログラム中の変数の値の変化を確認できる機能があります。この機能を使って、2つのセンサの値sen1, sen2が実際にどんな値を取っているのか、確認してみましょう。

① 39行目『sen1』をマウスで選択して右クリック！



④ 画面右上にある『Watch』ウィンドウにsen1, sen2が追加されているのを確認(*108)

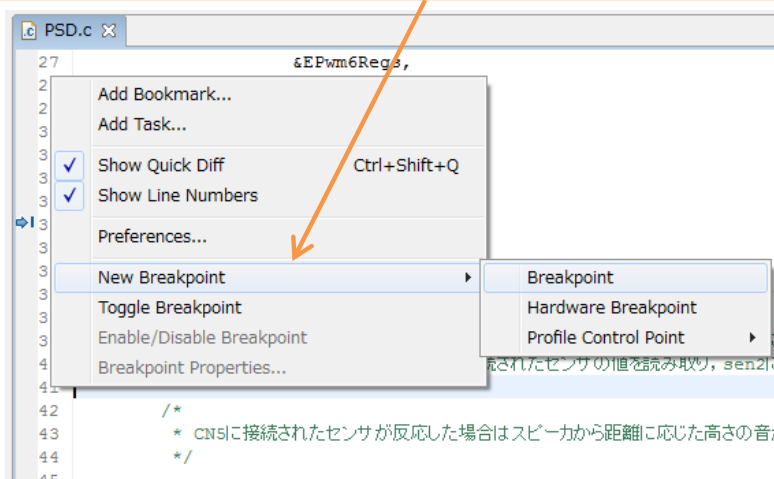
Name	Value	Address
sen1	37276	0x00008D19@Data
sen2	13348	0x00008D18@Data
<new>		



(*108)
Watch』ウィンドウがない場合

『Watch』ウィンドウがない場合は、左上の『View』>『Watch』をクリック

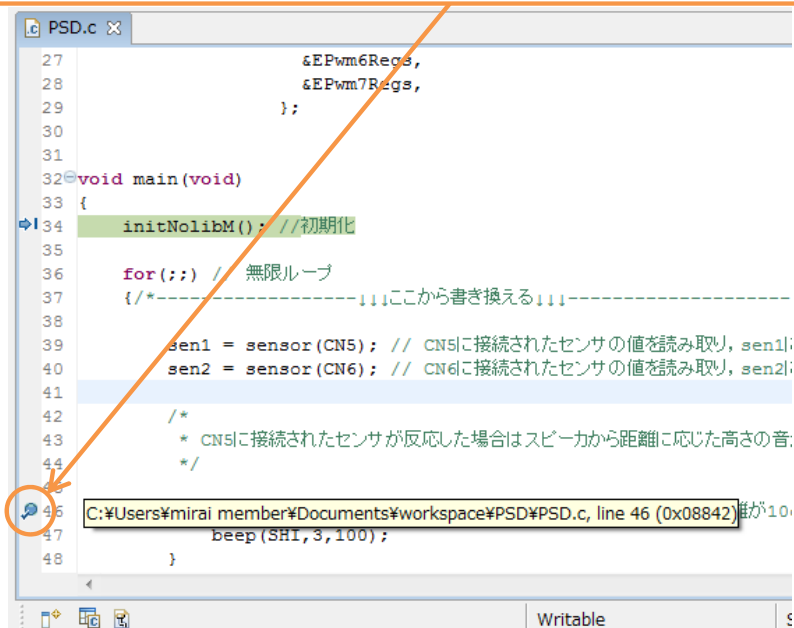
⑤ 41行目で右クリックして、『New Breakpoint』>『Breakpoint』を選択(*109)



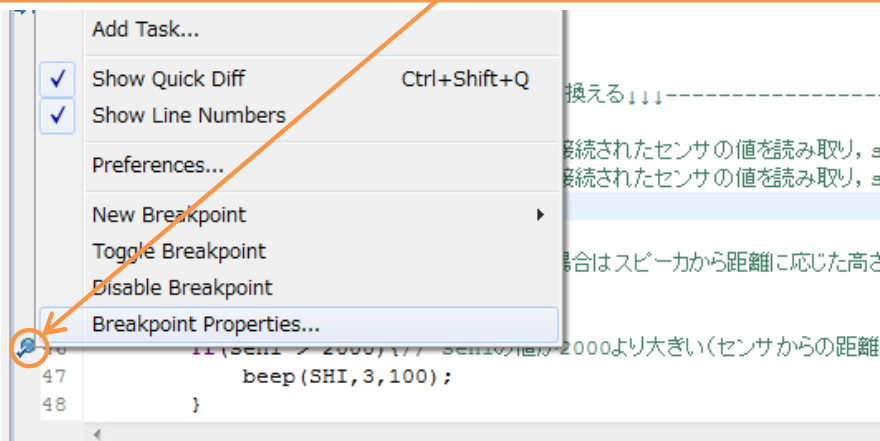
(*109)
Breakpoint(ブレイクポイント)

プログラムを実行すると、無限ループで処理がずっと行われるので、変数の値も常に変化してしまいます。変数の値を確認するために、処理をちょっと止めるのが、Breakpointです。

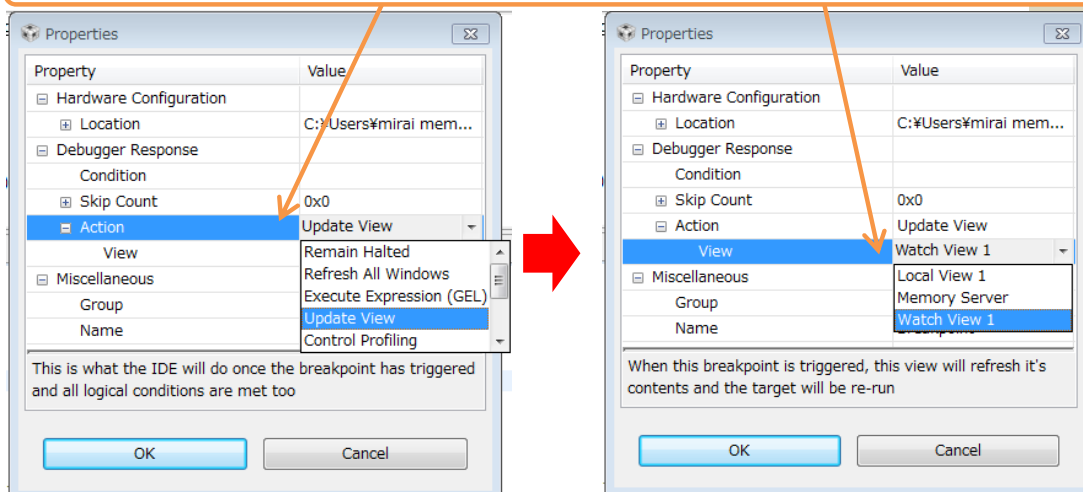
⑥ 46行目付近の行番号の左に、青いカギマークがついているのを確認



⑦ カギマークを右クリックし、『Breakpoint Properties...』を選択

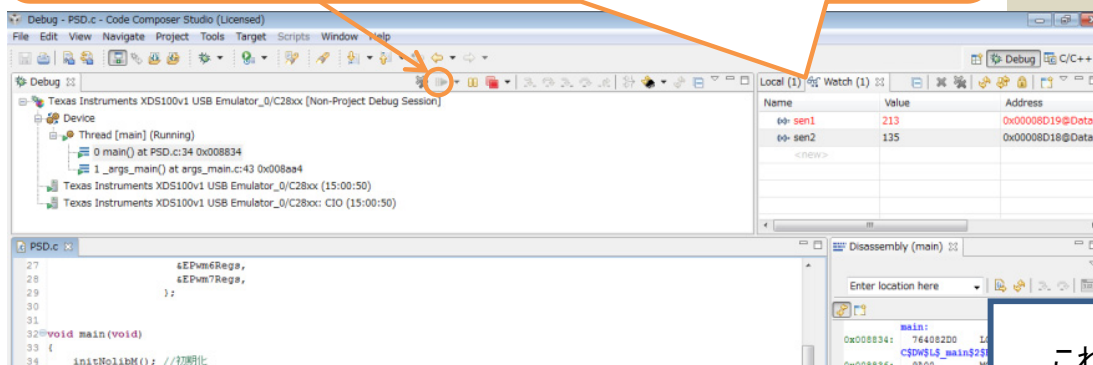


⑧ 『Action』>『Update View』を選択後、その下に出てくる『View』>『Watch View1』を選択



⑨ この状態で『Run』を行うと、『Watch』ウィンドウのsen1, sen2の値が更新される(*110)

Name	Value	Address
sen1	213	0x00008D19@Data
sen2	135	0x00008D18@Data



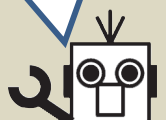
(*110)
トラブルシューティング

更新されない場合は、
① I/Oボードの電源が入っているか
② Breakpoint Propertiesの設定を確認

8-6 PSDチャレンジ

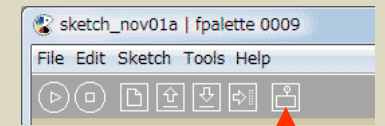
- ① 『Watch』で値を見ながら、センサからの距離に応じてどのくらいの値が出力されるかを確認しよう！
- ② センサに10cmよりも近づくと、出力される値がどうなるか確認しよう！ → データシートのグラフを見比べてみよう！
- ③ if([条件A] && [条件B])やif([条件A] || [条件B])を使って、自分なりのプログラムを書いてみよう！

これでロボットに必要なコトは一通り身につけることできたゾ！
次章では、これまでやってきたことを組み合わせてみよう！



8-7 f-palette IDEのプログラムの実行

これまでと同様に、5-4と同じ手順でサンプルプログラム『PSD』を実行してみましょう。CCSと同様に、センサのデータを確認することができます。f-palette IDEの「Serial Monitor」ボタン（ツールバーの右端）を押してください。下部のウィンドウに2つのセンサから送られてきたデータが流れていきます。



Serial Monitor

無限ループ

**sen1=sensor(A1);
sen2=sensor(A2);**
でA1、A2のセンサ値を取得

sen1、sen2の値をパソコンに送り、表示させるプログラム

A1のセンサの値**sen1**に応じてbeep音を変えるプログラム

これ以降は、A2のセンサの値**sen2**に応じてLEDを点滅させるプログラム

```
#include "fpalette_pin.h"
#include "palette.h"

int sen1, sen2;

void setup()
{
  palette_init();
  Serial.begin(9600);
}

void loop()
{
  sen1 = sensor(A1);
  sen2 = sensor(A2);
  /*
   *
   */
  Serial.print("sen1:");
  Serial.print(sen1);
  Serial.print(" sen2:");
  Serial.println(sen2);
  delay(10);

  if(sen1 > 2000){
    beep(SHI,D1,100);
  }
  else if(sen1 <= 2000 && sen1 >1800){
    beep(RA,D1,100);
  }
  else if(sen1 <= 1800 && sen1 > 1600){
    beep(SO,D1,100);
  }
  else if(sen1 <= 1600 && sen1 > 1400){
    beep(FA,D1,100);
  }
  else if(sen1 <= 1400 && sen1 > 1200){
    beep(MI,D1,100);
  }
  else if(sen1 <= 1200 && sen1 > 1000){
    beep(RE,D1,100);
  }
  else if(sen1 <= 1000 && sen1 > 900){
    beep(DO,D1,100);
  }
  else{
    ...
  }
  .
  .
}
```

sen1の値が2000より大きい場合、**【シ】**をCN11から100[ms]出す

sen1が1800より大きく2000以下の場合、**【ラ】**をCN11から100[ms]出す

sen1が1600より大きく1800以下の場合、**【ソ】**をCN11から100[ms]出す

sen1が1400より大きく1600以下の場合、**【ファ】**をCN11から100[ms]出す

sen1が1200より大きく1400以下の場合、**【ミ】**をCN11から100[ms]出す

sen1が1000より大きく1200以下の場合、**【レ】**をCN11から100[ms]出す

sen1が900より大きく1000以下の場合、**【ド】**をCN11から100[ms]出す

上の条件に当てはまらない(sen1の値が900以下)の場合、**何もしない**

if/elseによる条件分岐や、if([条件A] && [条件B])やif([条件A] || [条件B])による条件設定には慣れましたか？ これを使って、自分なりのプログラムを書いてみましょう。

12 秋葉原に出かけよう

概要

最近の秋葉原はアニメ、アイドルでも有名ですが、秋葉(アキバ)といえはやはり世界に誇る電気街です。せっかく日本にいるのですから一度はアキバにでかけてディープな世界をのぞいてみましょう。特にこれから電子工作を極めようとしている人は必須です。地図を片手に出かけましょう。

12-1 奥村研究員お薦めショップ

12-2 秋葉原マップ

12-1 奥村研究員お薦めショップ

秋月電子 [Map⑨]

黄色い取扱説明書が目印のこの店に行かなかったらモグリです。電子工作キット、マイコン、センサを豊富に取りそろえています。東大生が多数アルバイトをしていることでも有名。

購入した電子部品

LCD、LED、スピーカ、PSD、電池ボックス、焦電センサ、赤外線モジュール、アンプ、マイク、各種トランジスタ、74シリーズ、サーボモータ 他



秋月電子の店内



※ショップの情報はかわることがあります。事前の確認をお忘れなく。



東京都千代田区外神田1-8-3 野水ビル1F
03-3251-1779 (正午～18:00)
※月、木は電話対応は休み
月～土11:30～18:30 日祝祭11:00～18:00 無休

お気に入りの一品

多機能テスター
非常に低価格で使い勝手が良いです。軽いのでどこにいくにも持っていきます。



ACアダプタ

3.3V、5V、6V、12Vなど直流電源によく使用します。電池の代わりにどうぞ。(※モータも動かしたいときは出力電流Aが大きいものを使う)



千石電商 [Map⑩]

ロボット部品、マイコン、IC、LED、コネクタ、ケーブルなどなんでも幅広く取りそろえているアキバの代表的なお店。値段は他の店よりちょっと高いことがありますが、品ぞろえは素晴らしいです。

1号店で購入した電子部品

LED、基板、抵抗器、コンデンサ、ダイオードプーリー、三端子レギュレータ、ケーブル、モータドライバ、LED、7セグメントLED、コネクタ、スミチューブ 他

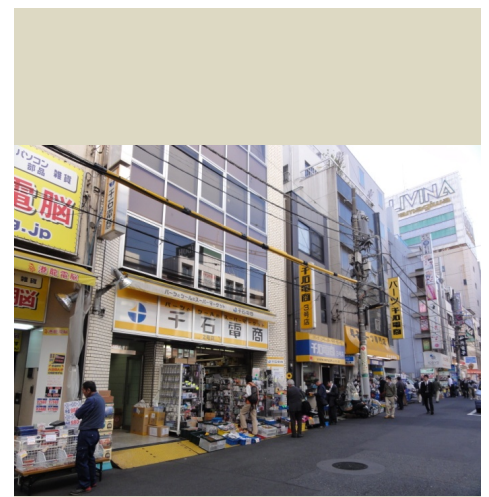


2号店で購入した電子部品 タミヤのキット、ギアボックス 他



3号店で購入した電子部品

モータ、ロボットオモチャ、ロット、ベルト、ギア、サーボモータ 他



本店 : 東京都千代田区外神田1-8-5
毛利ビル
日 10:00~19:00 月~木 10:30~19:00
金土 10:30~19:30 祝 10:00~(閉店時間は変更なし)

1号店から3号店まであります。
1号店が電子部品、2号店がロボットオモチャ、3号店が機械部品となっています。

山長通商 [Map②]

アキバに行ったら、まず「ガード下」。ヤマチョーと呼びます。コネクタなら間違いなくここ。

購入した電子部品

スイッチ、可変抵抗、7セグメントLED、コネクタ、モータ 他



東京都千代田区外神田1丁目14-2 ラジオセンター
03-3253-9389
10:00~18:30
年中無休

エスエス無線 [Map⑤]

アルミ部材などが豊富なので、ロボットをつくるときに大変便利なお店です。



東京都千代田区外神田1丁目10番11号
東京ラジオデパート2F
03-3251-7890
10:00~18:00
年中無休

小柳出電気商会(オヤイデ) [Map⑥]

ケーブル専門店。ケーブルのみです。カスタマイズケーブルもここで作ることができます。店頭以外にも在庫があるので、まずはお店の人に聞いてみましょう。



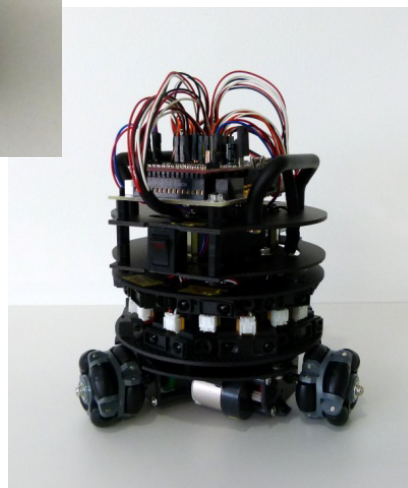
東京都千代田区外神田1-4-13
03-3253-9351
10:00~19:00(日定休、祝日は営業)

ヴィストンロボットセンター [Map⑭]

ロボット専門店。ロボットオモチャから研究用ロボットまで幅広く取りそろえています。ロボット部品も充実しています。

購入した部品

オムニホイール(全方位タイヤ)。タミヤのキット、ギアボックスもここで購入できます。



オムニホイールを使って作った全方位ロボット。製作期間1週間程度。

お気に入りの一品
絶縁布テープ、ビニールテープ。
ちょっと高いが、非常に巻きやすい！
本当によく使います。



東京都千代田区外神田1-9-9
内田ビル4F
TEL:03-3256-6676
10:30~19:00(水曜定休、年末年始)

フタバ産業 [Map⑬]

ラジコン専門店。ロボット部品に流用できるパーツが多数あります。
リチウムポリマー電池の種類も豊富です。

リチウムポリマーバッテリー

HallucIIのバッテリーもここで購入しています。



HallucII

西川電子部品[Map⑦]

ネジ専門店。ネジだらけです。スペーサー、工具、スミチューブも
ここ。



東京都千代田区神田須田町2-9 宮川ビル
1、2F
03-3255-9984



東京都千代田区外神田1-9-9
内田ビル1F
03-3253-6715
10:00~18:30(日・祭定休)

思わぬ掘り出し物がきつと見つかる店

あきばお〜 [Map⑰]

かなり安い！機能は！？自分の目で確かめにいきましょう。SDカード、CD-R、DVD-R、乾電池などの消耗品はかなり安くお勧めです。

購入したもの

デジタルノギス(1/100mm精度は怪しいが、これを使って全方位ロボット(P.4)を設計しました。)と精密ドライバーセット(滅多にみないトルクスと六角レンチの替えビット付き！！超レアもの。)。驚きの価格です。



東京都千代田区外神田3-1-12
03-3257-0235
月～金11:00～20:00
土日祝祭日10:30～19:30
店舗が沢山あるので、場所と営業時間を調べてから出かけましょう。

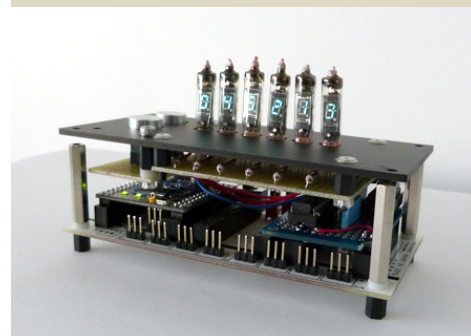
Webで購入したオモシログッズ

蛍光表示管
LEDがなかったころ使われていた表示器。扱いは少々厄介ですが、レトロ感がたまりません。



共立エレショップ
(商品の有無は直接お店に問い合わせてください)

その蛍光表示管を使って作ったストップウォッチ(作:奥村研究員)



Web shop

遠方の方や忙しい方にはウェブもお勧めです。

■電子部品を扱う店

秋月電子電商

<http://akizukidenshi.com/catalog/default.aspx>

店頭に置いていない部品もここで発見できます。

千石電商オンラインショップ

<http://www.sengoku.co.jp/>

マルツ

<http://www.marutsu.co.jp/>

良心的な価格で、レスポンスがとても早いです。スイッチ類が安いのでここで購入しています。

f-paletteの30ピンヘッダコネクタもここで売っています。

通販で購入しても領収書を希望すれば付けてくれます。

共立エレショップ

<http://eleshop.jp/shop/>

大阪日本橋に本店あります。

秋月、千石にはない、蛍光表示管など魅惑的なものも売っています。

通販で購入しても領収書を希望すれば付けてくれます。

digikey

<http://www.digikey.jp/>

世界最大規模の電子部品サイト、安い。半導体電子部品で欲しい部品があればまずはここを見るべし。
大量購入もここでもしています。

mouser electronics

<http://jp.mouser.com/>

半導体だけでなくコネクタ、リレーなどの部品もとりそろえます。大量購入ここでしています。
世界規模の販売店なので在庫、種類ともに圧倒的です。

RSコンポーネント

<http://jp.rs-online.com/web/>

モータ、工具、電子部品、サプライ品、、、いろいろ買えます。若干価格高いものもありますが在庫有れば18時までに注文すると翌日来ます。急用には良いです。

■電子キットを扱うお店

キット化した商品を扱うWEBショップ、ホビースト向け。電子部品から直接、自分で作る事が多いのであまり利用しませんが作るのが面倒で手っ取り早い事を好む人向け

浅草ギ研

<http://www.robotsfx.com/>

不思議なセンサだったりアクチュエータだったり売っています。
使い方などもサイトにのっています。
商品自体は千石電商でも扱っています。

ストロベリーリナックス

<http://strawberry-linux.com/>

目の付けどころが鋭い商品を取り揃えています。
sparkfun社商品買えます。sparkfun社は良心的な値段で手に入れ 難しいセンサなど取り付け易いように基板を付けて販売している会社です。
センサなどf-paletteに接続できるものも多いです。

スイッチサイエンス

<http://www.switch-science.com/>

arduino関連が主力商品のようです。
sparkfun社商品を多く扱います。

メカロボショップ

<http://www.mecharoboshop.com/>

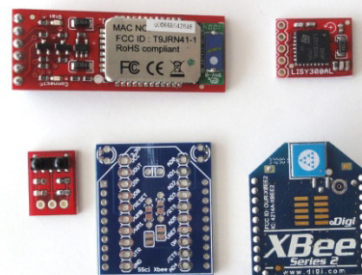
モータ、モータドライバ、ジャイロセンサなどをモーションに関連した商品を多く扱っています。

様々なスイッチ

ボタン、トグル、回転など、用途によってスイッチのかたち、サイズもかわります。fuRoではちょっとした時に使えるよう、普段からある程度の種類をストックしています。



Bluetoothモジュール
zigbeeモジュール
ジャイロセンサ
赤外線センサ
すべてf-paletteに接続ができます！

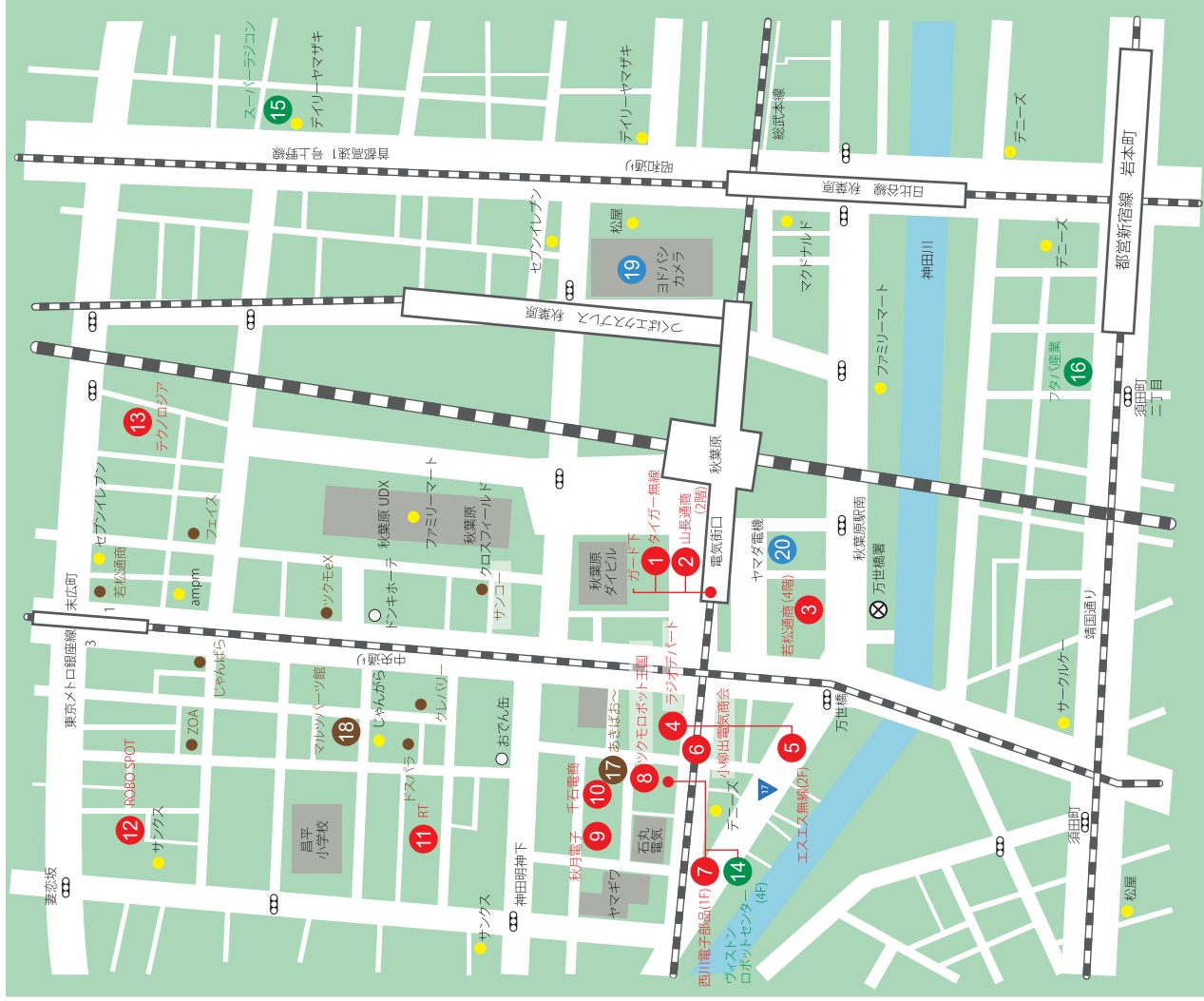


スイッチサイエンス
(商品の有無は直接お店に問い合わせてください)

秋葉原マップ

※お店の情報は変わることがあります。入れ替わりや店舗の営業時間など、変更がありましたら、教えてください。また、行く前にはウェブ等で調べてから行くことをおすすめします。

- ROBOT (電子パーツ・部材)
- ホビー
- PCパーツ
- 量販店
- 飲食店・コンビニ

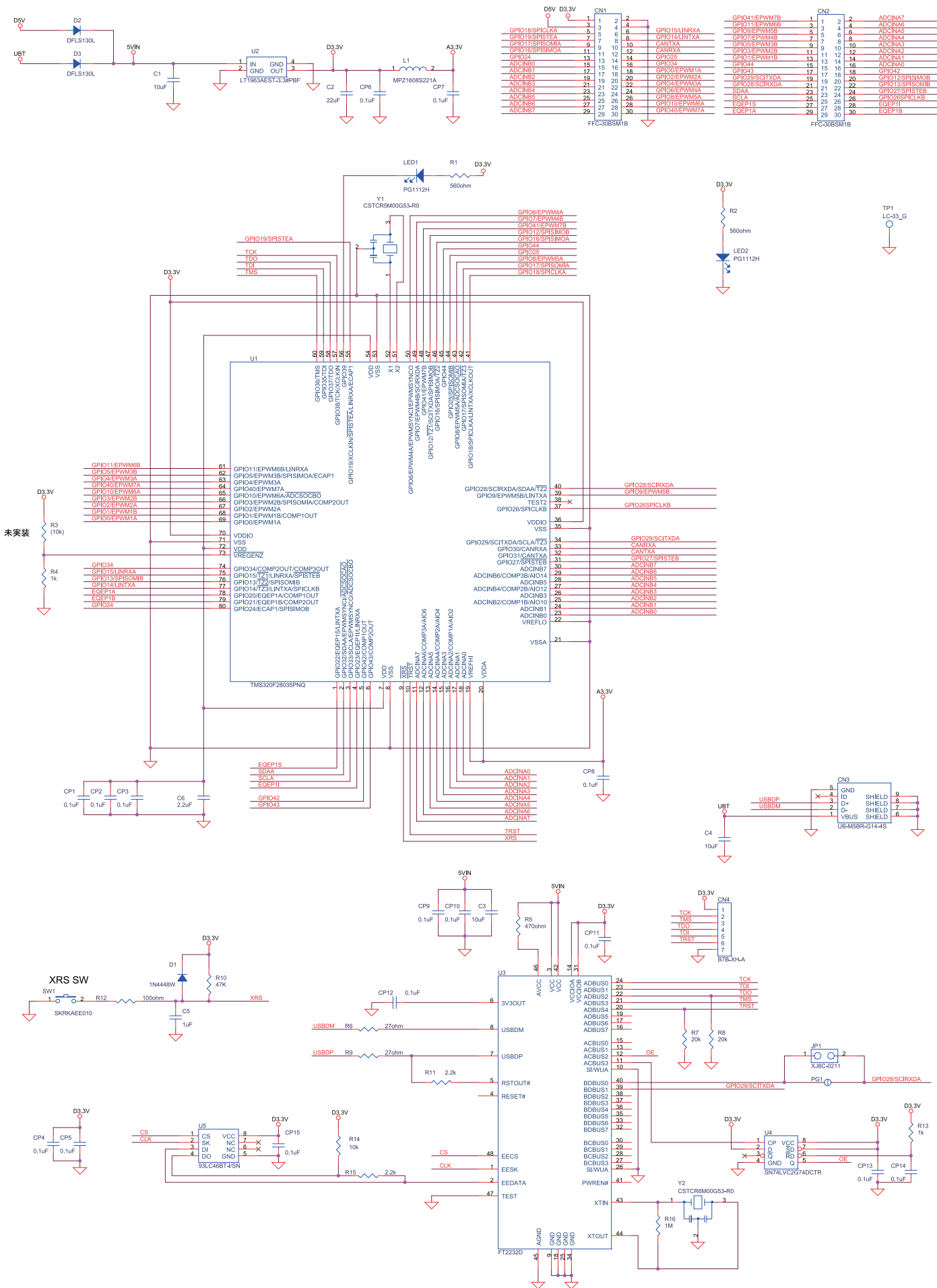


- 1 タイガー無線
取り扱い: ケーブル
千代田区外神田1-14-3 秋葉原電波会館1F
03-3251-6313
月～土 10:30～18:00 日祭 10:30～17:00
- 2 山長通商 (P.126参照)
東京都千代田区外神田1丁目14-2 ラジオセンター
03-3253-9389
10:00～18:30 (年中無休)
- 3 若松通商
東京都千代田区外神田4丁目7番3号 若松通商ビル4F
03-3251-8933
月～土 10:30～20:00 (年中無休)
- 4 ラジオデパート
東京都千代田区外神田1-10-11
<http://tokyoradiodepart.co.jp/tenpo/>
- 5 エスエス無線 (P.126参照)
ラジオデパート内
- 6 小柳出電気商会 (オヤイデ) (P.127参照)
03-3253-9351
東京都千代田区外神田1-4-13
10:00～19:00 (日曜定休、祝日は営業)
- 7 西川電子部品 (P.128参照)
03-3253-6715
東京都千代田区外神田1-9-9
10:00～18:30 (日祝 定休)
- 8 ソックモロボット王国
東京都千代田区外神田1-9-9
03-3251-0987
月～土 10:00～22:00 日祝 10:00～22:00
- 9 秋月電子 (P.124参照)
東京都千代田区外神田1-8-3 野水ビル1F
03-3251-1779 (正午～18:00)
※月、木は電話対応は休み
月～土 11:30～18:30 日祝 11:00～18:00
- 10 千石電商 (P.125参照)
本店: 東京都千代田区外神田1-8-5 毛利ビル
日 10:00～19:00 月～木 10:30～19:00 金土 10:30～19:30 祝 10:00～ (閉店時間は変更なし)
- 11 RT
東京都千代田区外神田3丁目2-13 山口ビル3F
平日 11:00～18:30 (土日祝定休)

- 12 ROBOSPOT
取り扱い: 近藤科学直営店、工作機器レンタルもあり
東京都千代田区外神田3-6-13 清田商会ビル1F
03-6421-6976
平日 14:00～19:00 土日 10:00～19:00
(水曜、第2、4火祝定休)
- 13 テクノロジア
取り扱い: ロボットオモチャ、f-palette販売店
東京都千代田区外神田4-12-9 アプローズ秋葉原1F
03-6206-8383
平日 11:00～20:00 (水曜定休) 土 10:30～20:00
日祝 10:30～19:00
- 14 ヴィストンロボットセンター (P.127参照)
東京都千代田区外神田1-9-9 内田ビル4F
TEL: 03-3256-6676
10:30～19:00 (水曜定休、年末年始)
- 15 スーパーラジオコン
取り扱い: ラジコン専門店。ロボット部品に流用できる
パーツ多数
東京都台東区台東1-10-7 ホーザンビル2・3F
03-5688-1414
月～土 11:00～21:00 日祝 10:00～20:00 (年中無休)
- 16 フタバ産業 (P.128参照)
東京都千代田区神田須田町2丁目9 宮川ビル
03-3255-9984
- 17 あきばお (P.129参照)
東京都千代田区外神田3-1-12
03-3257-4235
月～金 11:00～20:00 土日祝祭日 10:30～19:30
店舗が沢山あるので、場所と営業時間を調べてから出かけましょう。
- 18 マルツバーツ館
取り扱い: 秋葉原にはめずらしく部品が綺麗に並んでいる店。ただし商品が多すぎるのでネットがお薦め
東京都千代田区外神田3-10-10
03-5296-7802
月～金 11:00～20:00 (年中無休) 土日祝 10:30～20:00
<http://www.marutsu.co.jp/>
- 19 ヨドバシカメラ
東京都千代田区神田花岡町1-1
03-5209-1010
9:30～22:00
- 20 ヤマダ電機
東京都千代田区外神田1-15-8
03-5207-6711
10:00～22:00

付録 f-paletteボード 回路図・使用部品一覧

a-1. CPUボード回路図

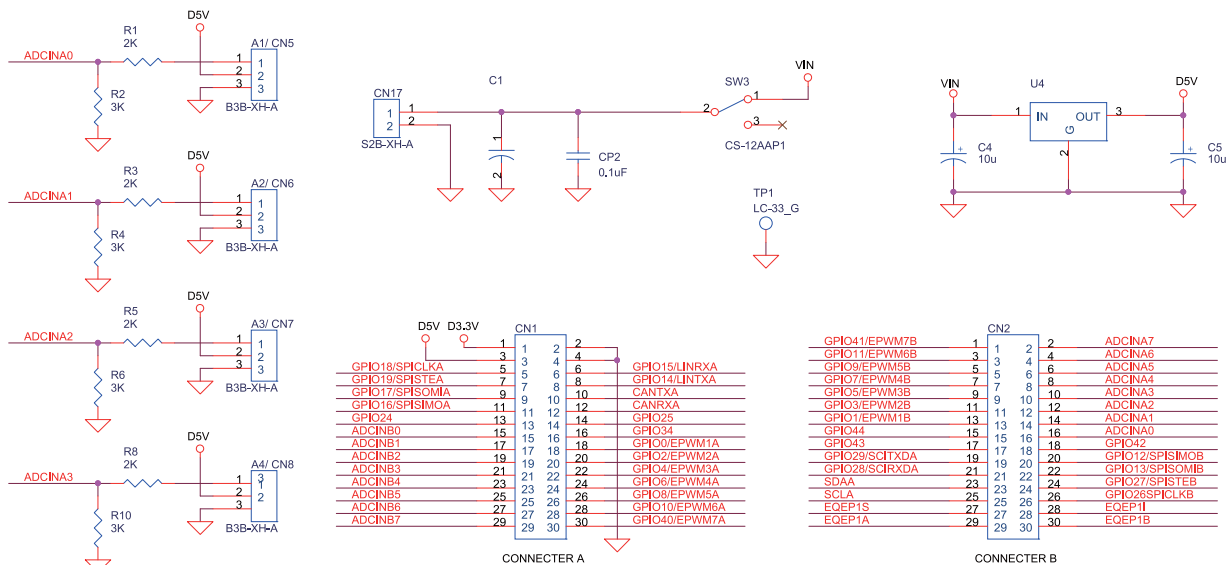


a-2. CPUボード部品表(基本の実装部品は記号及び使用個数を太字で記載)

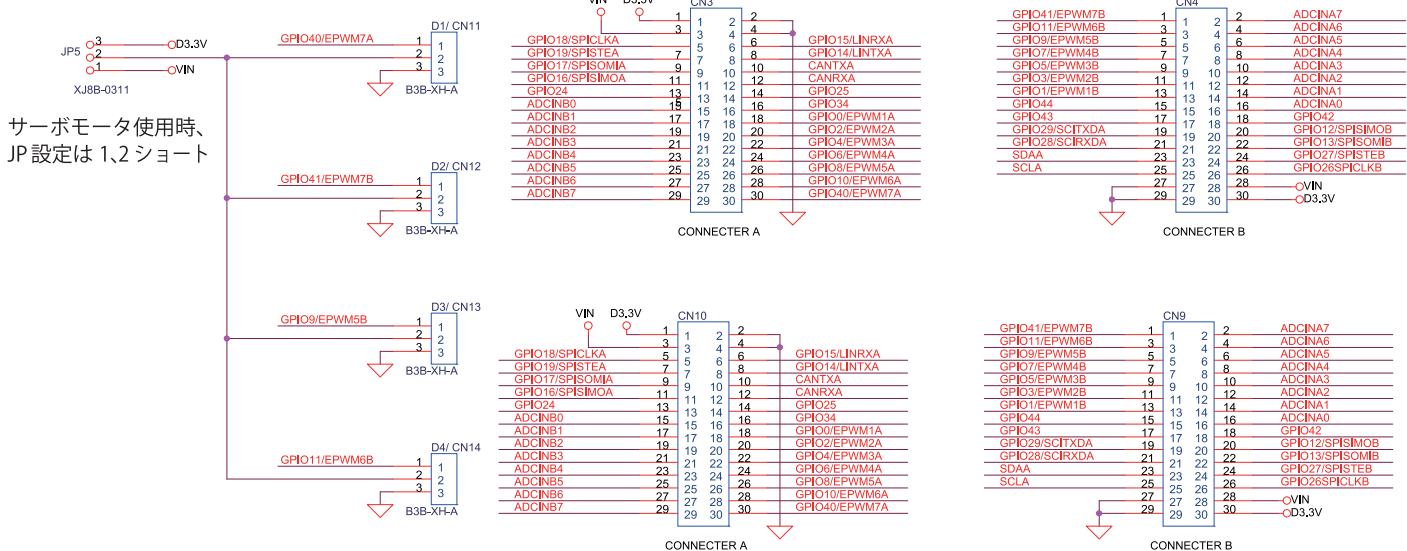
No.	品名	型番	メーカー	取付	個数	シルク印刷	備考
1	CPU	TMS320F28035PN	TI	面実装	1	U1	
2	レギュレータ	ZLDO1117G33TA	LT	面実装	1	U2	
3	Serial UART/FIFO IC	FT2232D	FTDI	面実装	1	U3	
4	ロジックIC	SN74LVC2G74DCTR	TI	面実装	1	U4	
5	EEPROM	93LC46BT-I/SN	マイクロチップ	面実装	1	U5	
6	発振子	CSTCR5M00G53-R0	村田	面実装	1	Y1	
7	発振子	CSTCR6M00G53-R0	村田	面実装	1	Y2	
8	フィルタ	MPZ1608S221A	TDK	面実装	1	L1	
9	ダイオード	1N4448W	DIODES	面実装	1	D1	
10	ショットキーダイオード	DFLS130L	DIODES	面実装	2	D2, D3	
11	LED	PG1112H	STANLEY	面実装	2	LED1, POWER	
12	チップ抵抗	RK73B1ETTD270J (27Ω)	KOA	面実装	2	R6, R9	
13	チップ抵抗	RK73B1ETTD101J (100Ω)	KOA	面実装	1	R12	
14	チップ抵抗	RK73B1ETTD471J (470Ω)	KOA	面実装	1	R5	
15	チップ抵抗	RK73B1ETTD561J (560Ω)	KOA	面実装	2	R1, R2	
16	チップ抵抗	RK73B1ETTD102J (1KΩ)	KOA	面実装	2	R4, R13	
17	チップ抵抗	RK73B1ETTD222J (2.2KΩ)	KOA	面実装	2	R11, R15	
18	チップ抵抗	RK73B1ETTD103J (10KΩ)	KOA	面実装	1	R14	
19	チップ抵抗	RK73B1ETTD203J (20KΩ)	KOA	面実装	2	R7, R8	
20	チップ抵抗	RK73B1ETTD473J (47KΩ)	KOA	面実装	1	R10	
21	チップ抵抗	RK73B1ETTD105J (1MΩ)	KOA	面実装	1	R16	
22	チップコンデンサ	GRM155F11E104ZA01B (0.1μF)	村田	面実装	15	CP1-15	
23	チップコンデンサ	GRM21BB11A105KA01K (1μF)	村田	面実装	1	C5	
24	チップコンデンサ	GRM21BB31A106KE18L (10μF)	村田	面実装	3	C1, C3, C4	
25	チップコンデンサ	GRM31CF11A226ZE01K (22μF)	村田	面実装	1	C2	
26	チップコンデンサ	GRM316B11A225KA01B (2.2μF)	村田	面実装	1	C6	
27	USBコネクタ	UB-M5BR-G14-4S	JST	面実装	1	CN3	
28	30(2*15)ピンヘッダ	(2.54mmピッチ)	-	DIP	2	CN1, CN2	
29	スイッチ	SKRKAEE010	アルプス	面実装	1	SW1	マイコンのリセットスイッチ
30	1*2ピンヘッダ	(2.54mmピッチ)	-	DIP	1	JP1	PG1の配線カット時の再接続用
31	ジャンパピン	(2.54mmピッチ)	-	—	1	JP1	PG1の配線カット時の再接続用
32	テスト端子	LC-33_G	MAC8	DIP	1	TP1	
33	チップ抵抗	RK73B1ETTD103J (10KΩ)	KOA	面実装	1	R3	R3/R4のどちらかを実装することで内部電圧レギュレータの不使用/使用を選択
34	7pinコネクタ オス	B7B-XH-A	JST	DIP	1	CN4	JTAG(デバッグ用シリアル通信)用

b-1. I/Oボード回路図

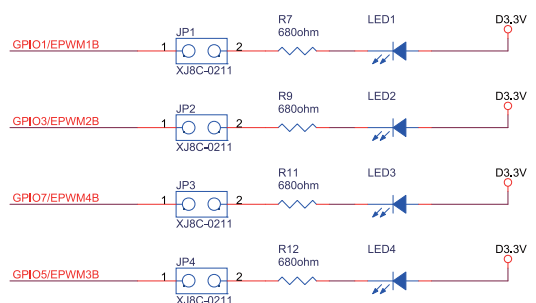
ADC コネクタ



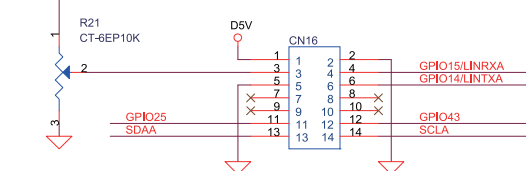
GPIO コネクタ



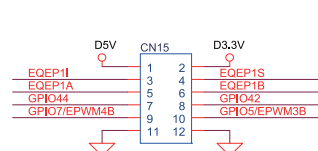
LED



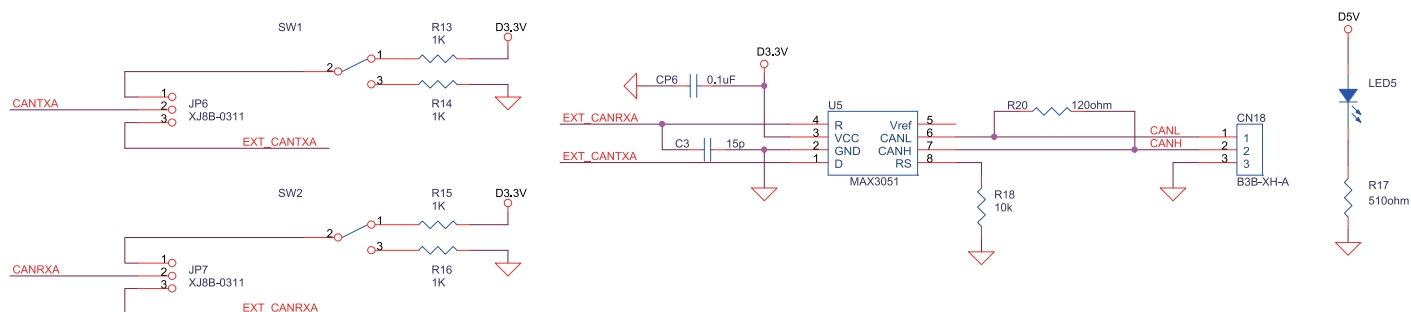
LCD 用コネクタ



エンコーダコネクタ用



電源確認 LED

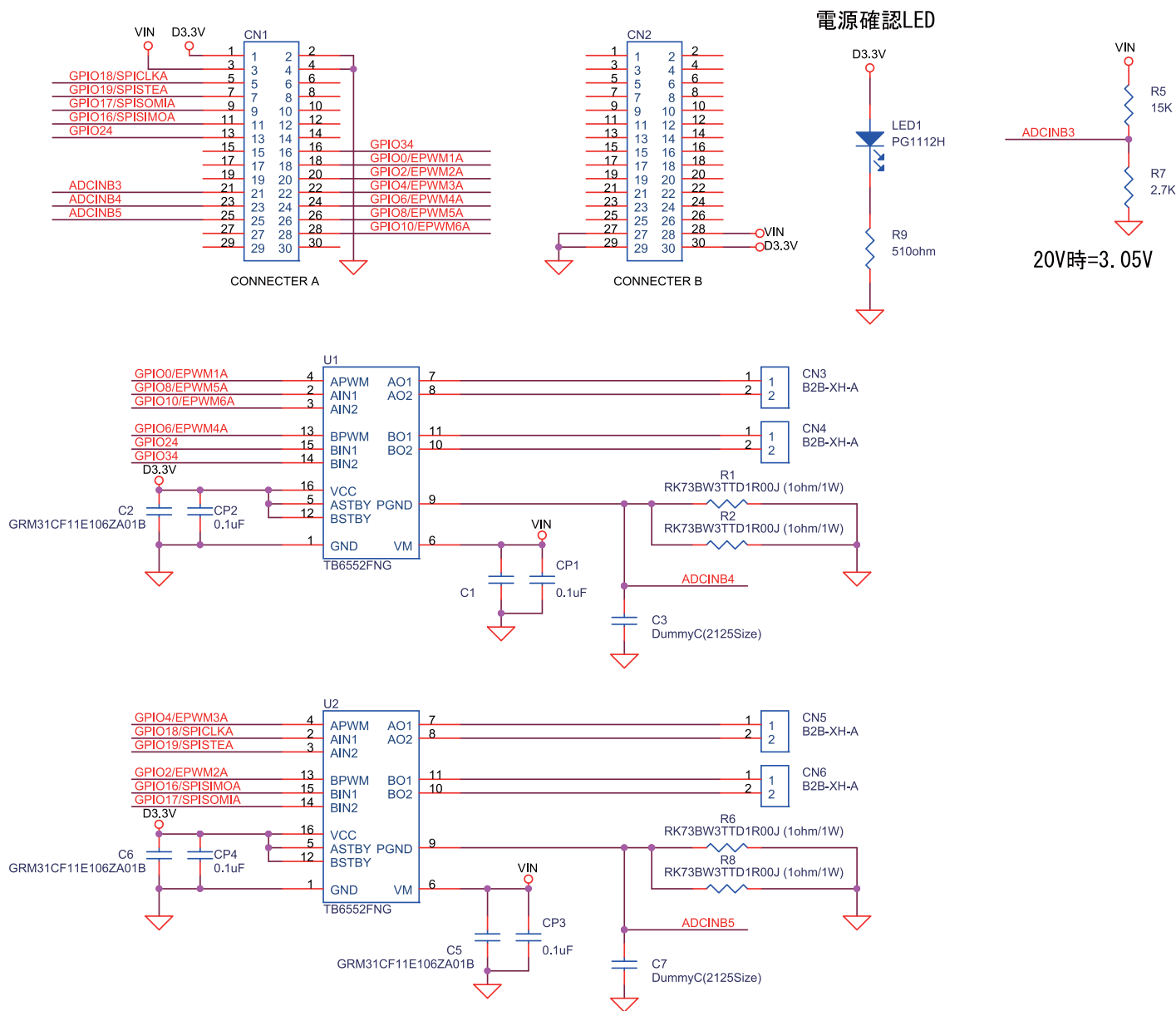


トルグスイッチ使用時は 1、2 ショート
CAN 使用時は 2、3 ショート

b-2. I/Oボード部品表(基本の実装部品は記号及び使用個数を太字で記載)

No.	品名	型番	メーカー	個数	記号(シルク印刷)	取付	購入先	備考
1	3端子レギュレータ	(5V/1.5A)		1	U4	DIP		
2	赤色LED	OSR5JA3E34B	OptoSupply	1	LED5	DIP	秋月	
3	黄緑LED	GL3JE402B0P1	SHARP	4	LED1, LED2, LED3, LED4	DIP	秋月	
4	炭素皮膜リード抵抗	1/6W, 510Ω	—	1	R17	DIP	秋月	
5	炭素皮膜リード抵抗	1/6W, 680Ω	—	4	R7, R9, R11, R12	DIP	秋月電子	
6	炭素皮膜リード抵抗	1/6W, 1kΩ	—	2(4)	R13, R14, R15, R16	DIP	秋月電子	SW2使用: R15, 16を実装
7	炭素皮膜リード抵抗	1/6W, 2kΩ	—	2(4)	R1, R3, R5, R8	DIP	秋月電子	A3使用: R5, R6を実装
8	炭素皮膜リード抵抗	1/6W, 3kΩ	—	2(4)	R2, R4, R6, R10	DIP	秋月電子	A4使用: R8, R10を実装
9	アルミ電解コンデンサ	140-MLRL35V22-RC (22u/35V)	Xicon	1	C1	DIP	Mouser	廃番
10	積層セラミックコンデンサ	0.1uF/50V	Supertech Electronic	1(2)	CP2, CP6	DIP	秋月電子	CAN使用: CP6を実装
11	アルミ電解コンデンサ	EKZE500ELL100ME07D (10uF/50V)	United Chemi-Con	2	C4, C5	DIP	Mouser	
12	2*8ピンヘッダ	(2.54mmピッチ)	—	4	JP1, JP2, JP3, JP4	DIP	秋月電子	
13	1*3ピンヘッダ	(2.54mmピッチ)	—	2(3)	JP5, JP6, JP7	DIP	秋月電子	SW2使用: JP7を実装
14	ジャンパピン	(2.54mmピッチ)	—	6(7)	JP1, JP2, JP3, JP4, JP5, JP6, JP7	—	秋月電子	SW2使用: JP7を実装
15	トグルスイッチ	200C-WMSP-1-T2-B4-M2	Taiway	1(2)	SW1, SW2	DIP		SW2使用の場合は実装
16	スライドスイッチ	CS-12AAP1	日本開閉器	1	SW3	DIP	マルツ	
17	30(2*15)ピンソケット	21602X15GSE	Linkman	4(6)	CN1, CN2, CN3, CN4, CN9, CN10	DIP	マルツ	モータドライバ2枚使用の場合、CN9, CN10を実装
18	1*3ピンヘッダ	(2.54mmピッチ)	—	4(9)	A1/CN5, A2/CN6, A3/CN7, A4/CN8, D1/CN11, D2/CN12, D3/CN13, D4/CN14, CAN/CN18	DIP	秋月電子	A3, A4, D2, D3, CANを使用の場合はハンダ付け
19	2ピン電源ソケット	S2B-XH-A	JST	1	CN17	DIP	JST	
20	3.3V CAN トランシーバ		TI	1	U5	面実装	Mouser	CAN使用の場合に実装
21	炭素皮膜リード抵抗	120Ω	—	1	R20	DIP	秋月電子	CAN使用の場合に実装
22	炭素皮膜リード抵抗	10KΩ	—	1	R18	DIP	秋月電子	CAN使用の場合に実装
23	サーメットリマ	CT-6EP10K	COPAL	1	R21	DIP	マルツ	LCD使用の場合に実装
24	積層セラミックコンデンサ	15pF/50V	—	1	C3	DIP	Mouser	CAN使用の場合に実装
25	テスト端子	LC-33_G (黒)	MAC8	1	TP1	DIP		
26	12(2*6)ピンソケット	(2.54mmピッチ)	—	1	CN15	DIP	秋月電子	EQEP使用の場合に実装
27	14(2*7)ピンソケット	(2.54mmピッチ)	—	1	CN12	DIP	秋月電子	LCD使用の場合に実装

c-1. モータドライバボード回路図



c-2. モータドライバボード部品表(基本の実装部品は記号及び使用個数を太字で記載)

No.	品名	型番	メーカ	個数	記号(シルク印刷)	取付	備考
1	モータドライバ	TB6552FNG	東芝	2	U1, U2	面実装	
2	LED	PG1112H	STANLEY	1	LED1	面実装	
3	チップ抵抗	RK73B1ETTD511J (510Ω)	KOA	1	R9	面実装	
4	チップ抵抗	RK73B1ETTD272J (2.7KΩ)	KOA	1	R7	面実装	
5	チップ抵抗	RK73B1ETTD153J (15KΩ)	KOA	1	R5	面実装	
6	チップ抵抗	RK73BWA3TTD1R00J (1ohm/1W)	KOA	4	R1, R2, R6, R8	面実装	
7	チップコンデンサ	GRM155F11E104ZA01B (0.1uF)	村田	4	CP1, CP2, CP3, CP4	面実装	
8	チップコンデンサ	GRM31CF11E106ZA01B (10uF)	村田	4	C1, C2, C5, C6	面実装	
9	1*2ピンヘッダ	(2.54mmピッチ)	JST	4	CN3, CN4, CN5, CN6	DIP	
10	30(2*15)ピンヘッダ	(2.54mmピッチ)	本多	2	CN1,CN2	DIP	
11	チップコンデンサ	-	-	2	C3, C7	面実装	未実装