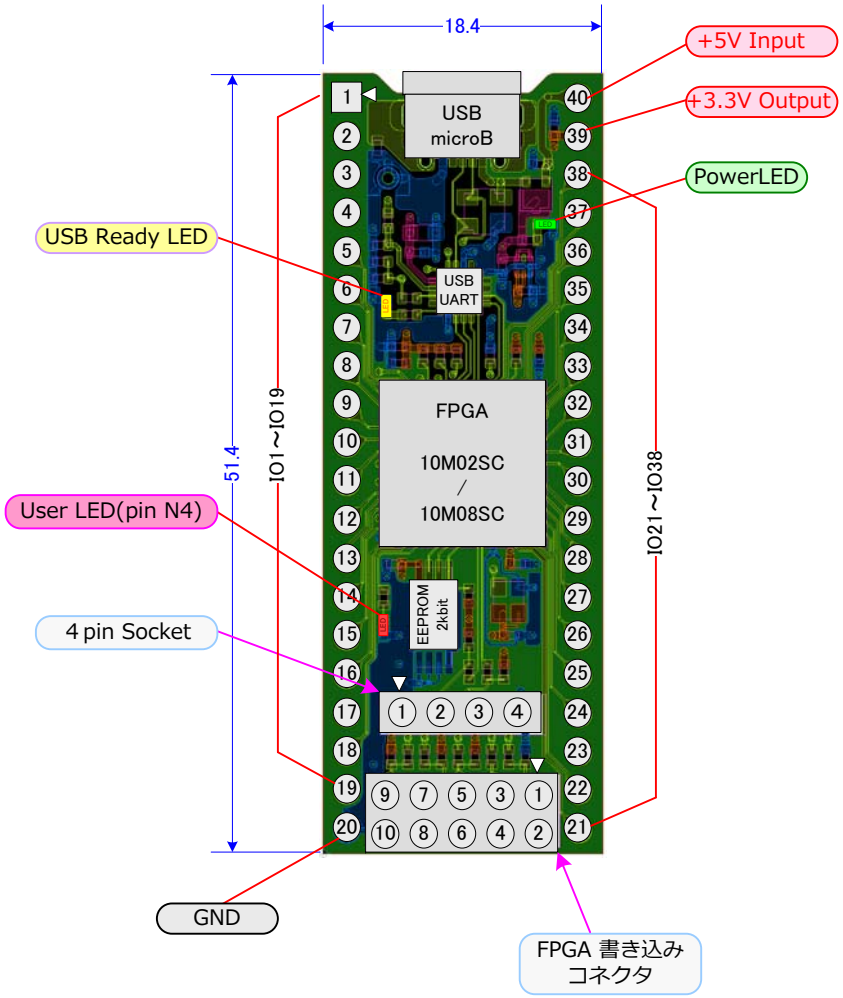


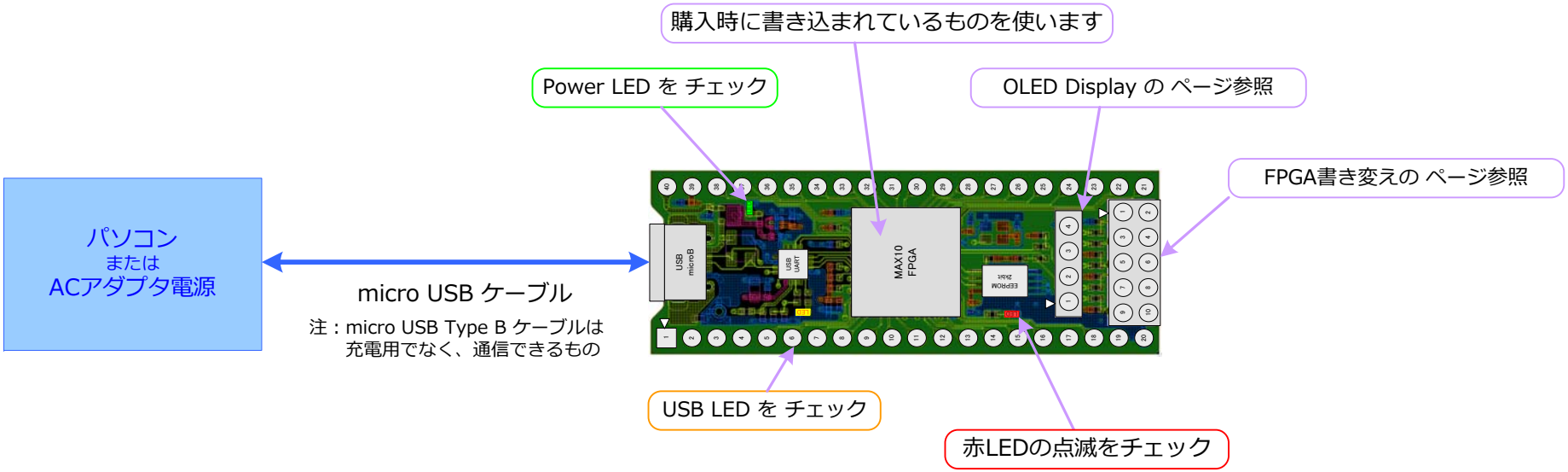
Features

- ☆ シンプルで超低コストのFPGAボード
- ☆ フラッシュメモリを内蔵したFPGA 10M02SCU / 10M08SCUを搭載
- ☆ 使いやすい 2.54mmピッチの 40ピンICソケット互換サイズ
- ☆ USB-UART変換IC搭載で、PCからFPGAのデバッグに便利
- ☆ PCからFPGAを制御してシステムを作るのにも有効
- ☆ 不揮発性メモリ(EEPROM 2kbit) 搭載で、ちょっとした情報が記憶可能
- ☆ ４ピンコネクタ (I²C interface) には 128 x 32 pixel のディスプレイ接続可
- ☆ FPGAへの書き込みは USB Brasterやその互換品が使える。
- ☆ 用途に応じて iGf02(2000LC) iGf08(8000LC)が選択できる。
- ☆ 購入時のデモ実行ファイルなど Intel開発ツール用の環境データ付属
- ☆ 上記のソースコード(Verilog)付属
  - ☆ 使用される場合は自己責任でお願いします。
- ☆ 開発用ツール Intel Quartus Prime が必要です。（無償バージョン）
  - ☆ このボードは 無償バージョンで開発できます。
- ☆ FPGA書き変えには USB Blaster が必要です。（ 11ページ参照）
- ☆ micro USB (TypeB) ケーブルは 別途 お求めください。

Applications

- ☆ FPGA 学習、実験
- ☆ 製品試作
- ☆ 小ロット製品
- ☆ 趣味

最初に micro USB ケーブルをつなぎ、基板上のLEDが点滅することを確認しよう。



基板上のLEDが点滅すれば、FPGAに書き込まれている回路は正常に動作しています。

micro B の形状

いろいろなタイプがあるので、写真を載せておきます。

100円ショップ(セリア）に売ってた巻き取り型 ケーブル

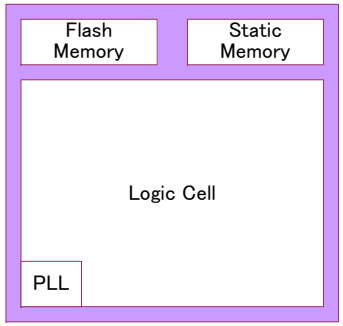
FPGAとMPUの違い

☆ FPGAは Field Programmable Gate Array の略
つまり書き換え可能なハードウェア回路ということです。

☆ FPGAの構成

メモリーは専用になっているが、
それ以外はロジックセルが並んでいる
だけで、極めてシンプル。

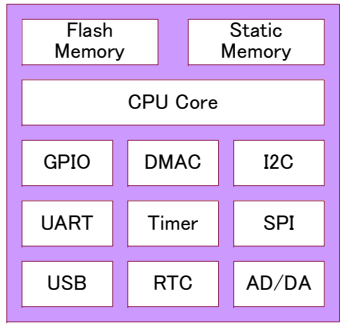
高機能版には CPU、高速シリアライザー、
ADCなどが専用化されているものもある。



PLLは高速クロックを生成するもの。
50MHzの入力から 200MHzを作ったり

☆ MPUは Micro Processing Unit の略
こちらもプログラムを書き換えることができます。

☆ MPUの構成



CPUコアの周りに、いろいろな
インターフェースが付いている。
その種類によって、多くの種類がある。

外部インターフェース

自作するかライブラリーを組み込まなければならない。
ライブラリーにはオープンソース、フリーライセンスのものもある。

組み込まれているので、使用するだけで良い。
数が足らなくなったり、特殊なものは対応できない。

開発の難易度

全てが同時に動いてしまうので、頭脳が破綻！？
シーケンシャル動作をさせるには、そのための論理が必要。

並列動作をしないので、考えやすい。

メリット

全てが同時に動かせることは、たとえば 100MHzで動作させた場合
全ての論理が 100MHzで同時動作する・・・極めて高速処理
たとえば、X = (A+B) * (C-D) - ・・・ これが 1クロックで完了
実際には ロジック遅延があるので、工夫が必要となるが

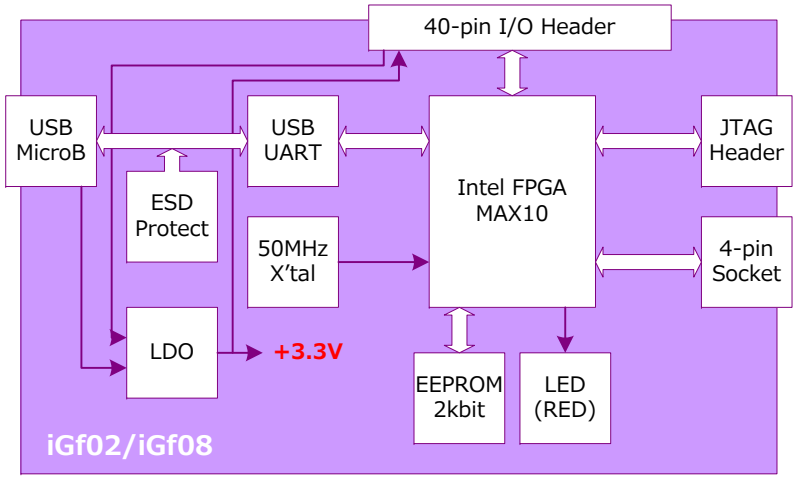
RT-OSなどのツールなどもあり、複雑な処理には最適。

好みの問題

0 からスクラッチで作りたい人向き、CPUも作れるヨ！
（人の作ったものを理解するのが面倒な人？）

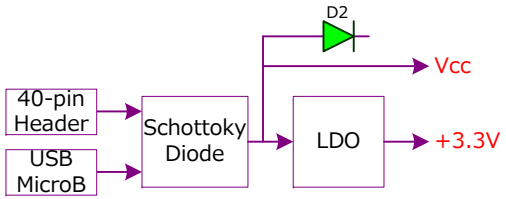
環境が揃っており、取り組みやすい。

Block Diagram



1. LDO (Low Drop Out Regulator)

USBコネクタからの電源と 40-pin Header の 40-pinからの電源がダイオードでオワされ LDOに入ります。

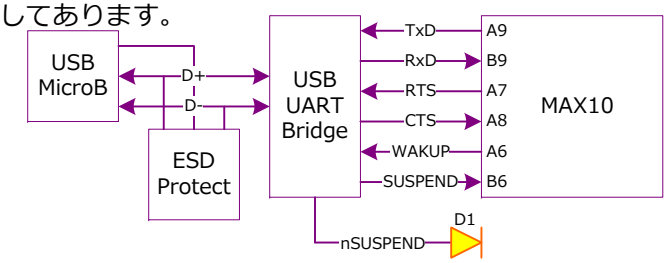


したがって、USBコネクタに接続するだけで、動作可能になります。
このLDOから 40-pin Headerの 39 pinに +3.3Vが出力されます。
（+3.3Vの 外部で使用できる電流容量は 100mA程度まで）
電源が入っているとき D2(Green LED)が点灯します。

2. USB to UART (CP2102N)

USB-UART 変換IC(CP2102N SILICON LABS)を通して、PCなどから FPGAをアクセスするために用意してあります。

USBがアクティブのとき、D1(Yellow LED)が点灯します。



デバック時に FPGA内のモジュールに設定を与えたりするときに役立ちます。
実動作時にも使用することで、フレキシビリティが向上します。

3. 40-pin I/O Header

40-pin 600-mil IC socket 互換のサイズで配置されたパッドに　FPGAの信号を入出力できます。
ピンアサインは 後述の "*iGf02 & iGf08 コネクタのピン配列*" を参照。
パッド間隔は 2.54 mm (0.1 inch) および 15.24 mm (0.6 inch)です。

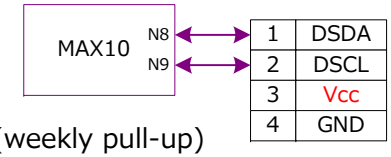
4. JTAG Header

10-pin Header に Intel および その互換の USB Blaster を接続します。
1 pin マーク（▲）に注意して、接続してください。

5. 4-pin Socket

この 4-pin Socket は、Demo用の OLEDを接続するためのものですが、汎用の I/Oとしても使用できます。

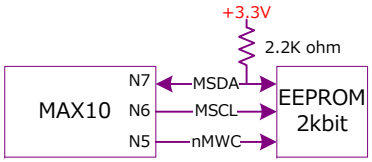
Vccは +5V 入力から、ダイオードを通して出力されるため、0～0.5V程度の電圧降下が発生します。



出力は 10 K ohmでプリアップされてますが、(weekly pull-up) デバイス側で、本来のプルアップ抵抗が必要です。
OLEDモジュールは GND(4pin側)が ■のランドになっていますので、注意してください。

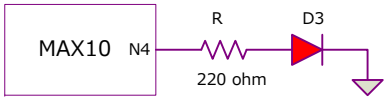
6. EEPROM (M24C02)

2Kbit の EEPROM(M24C02 STmicro)が接続されています。MSDAは双方向ですが、MSCLはプルアップ抵抗はありませんので注意してください。



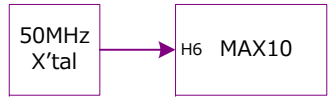
7. LED (Red)

赤色の LEDが接続されています。
明るさは PWM制御などで調整できます。



8. Clock (50 MHz crystal oscillator)

発振器は 50 MHz / ±50 ppm です。
FPGAのグローバルクロックピン(CLK1p)に接続。



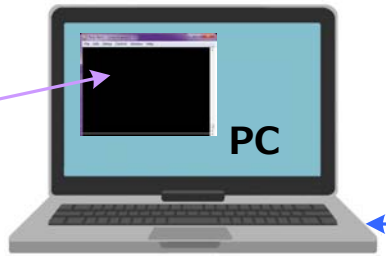
9. Reset

nReset 信号は 10 K ohm の抵抗でプルアップされています。HDLの Reset信号として使ってください。
外部リセットが必要なときは 40 pin コネクタからの信号を使用してください。

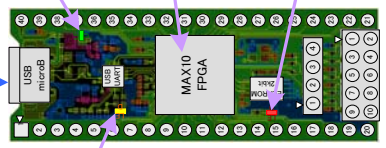
PCから基板上のLEDを制御してみる

ターミナルソフトを立ち上げて
テキストを打ち込みむことで
FPGA内のレジスタを書き変えて
デバッグ

☆ ターミナルソフトは TeraTermなどの
フリーソフトが使えます。



micro USB
ケーブル



購入時に書き込まれているものを使います

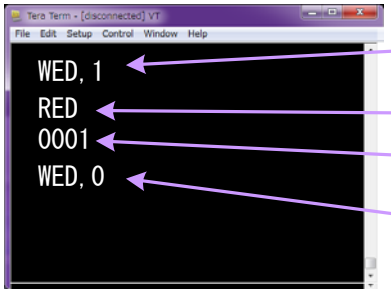
Power LED を チェック

基板上のLEDの点灯をテストしてみます

USB LED を チェック

UART 設定値

Baud rate : 115200
Data : 8 bit
Prity : none
Stop : 1 bit
Flow control : none



LEDは 点滅しています。

ED番地のレジスタに 0001をライト → LED 点灯

ED番地のレジスタからリード

ED番地内容が表示される

ED番地のレジスタに 0000をライト → LED 消灯

UARTと接続するための、サンプルソースが
付いています。（Verilog)
（書き込んであるもの全てのソース付）

☆ このように、なにかの設定を変えたりして機能を確かめたいとき、とても便利です。

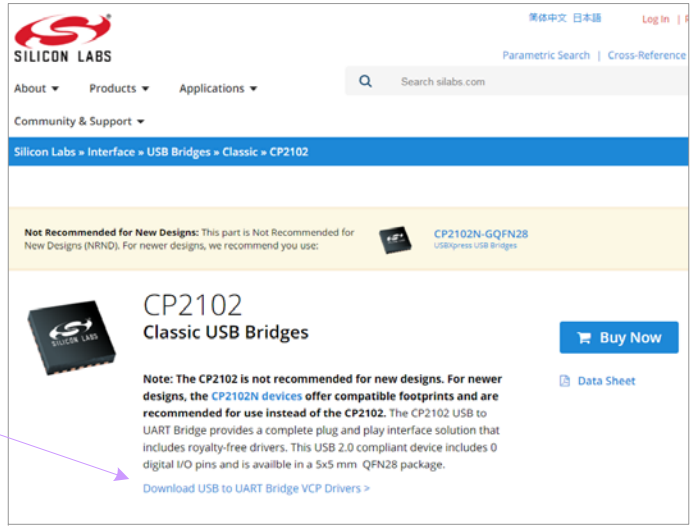
USB-UART 変換チップのドライバ インストール

☆ PCと接続するためには、USB-UART変換チップ(CP2102)のドライバをダウンロードする必要があります。

Silicon LabsのHPから CP2102へ行き、この場所からダウンロードします。

ダウンロードファイルを選択するページに行ったら、環境に合ったファイルをダウンロードします。

zip圧縮ファイルを解凍してから exeファイルを実行してインストールします。



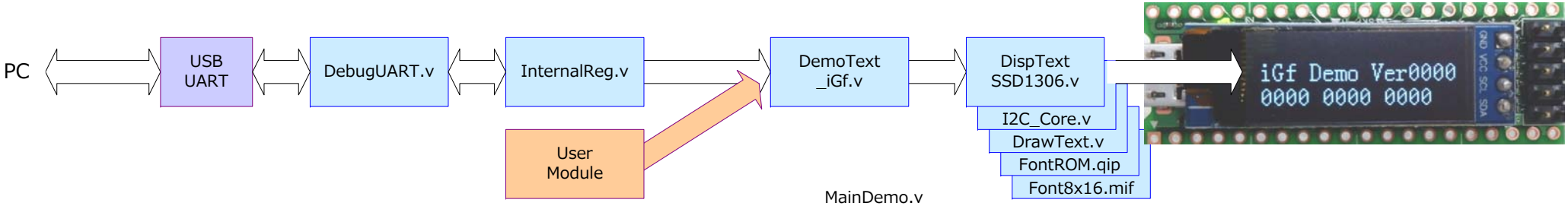
☆ おまけの “OLED 128x32 pixel”が付いています。（おまけですので、品質保証は致しかねます。破損してしまったときなどは、通販サイトなどでお求めください。）

OLEDボードは iGf ボードの 4pinコネクタに差してください。　（4pinコネクタだけで固定され、取り付け精度も高くないことを理解してご使用ください。）

USBを通して、各状態を確認できるのですが、常に表示しておきたいときや、高速で変化するときなどは、このようなディスプレイに出しておくのが便利です。

購入時には 16x8 dot のフォントが 16文字 2行 表示されます。PCのターミナルから データを書くと 2行目に文字を出すことができます。（数値のみ 16進数 0～F）

このPCから書き込まれたレジスタの代わりに、表示したいデータをHDLで作れば自由に表示できるようになります。



フォントデータは多くのLC（ロジックセル）を使ってしまいます。
MAX10-FPGAは、２つのフラッシュメモリ CFMとUFMを内蔵しています。
Demo では、フォントデータを UFMに書き込みました。

iGf02では Demo で使用しているLCは 65%程度で、残り 30%程度です。
しかし、初めて使用される方には十分な量だと思います。
物足りない方は、iGf08を購入してください。

MainDemo.v の ここを
表示したいデータに書き変える。
16bitの４桁の データを間隔を
空けて ３ヶ表示できます。

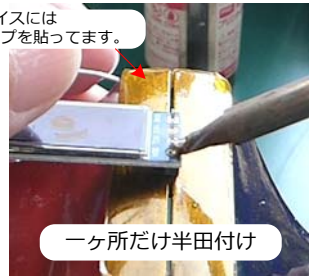
ASCII文字を表示したいときは
DemoText_iGf.v を
書き変えましょう。

MainDemo.v

```
DemoText_iGf DemoText_iGf_(
    .Reset          ( Reset         ),  // i Reset
    .Clock          ( CLK_50M       ),  // i Clock
    .CLKEN1ms       ( CLKEN1ms      ),  // i Clock Enable 1msec
    .InitialFlag    ( InitialFlag   ),  // i Initialize Status Flag
    .DispReg0       ( DispReg0      ),  // i from InternalReg
    .DispReg1       ( DispReg1      ),  // i from InternalReg
    .DispReg2       ( DispReg2      ),  // i from InternalReg
    .Xposi          ( DemoXposi     ),  // o X position
    .Yposi          ( DemoYposi     ),  // o Y position
    .FontCode       ( DemoFontCode  ),  // o ASCII Font Data
    .FontWr         ( DemoFontWrite ),  // o ASCII Font Write
    .WrReady        ( FontWrReady   ),  // i FontCode & Position Write Ready
    .DemoFlag       ( DemoFlag      )   // o Demo Access Status Flag
);
```

半田付け

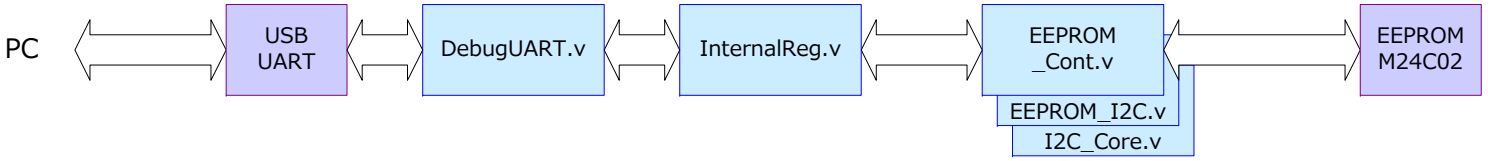
OLEDのモジュールは半田付けされておりません。付属の短い方の コネクタの 半田付けしてください。ピンの長い方が半田付けする側で、短い方が差し込む側です。



半田ごての使い方ですが、半田をピンに付けるのではなく、ピンとランドを熱したところに半田を乗せます。
ただ、こて先にも半田を付けておいたほうがいいようです。

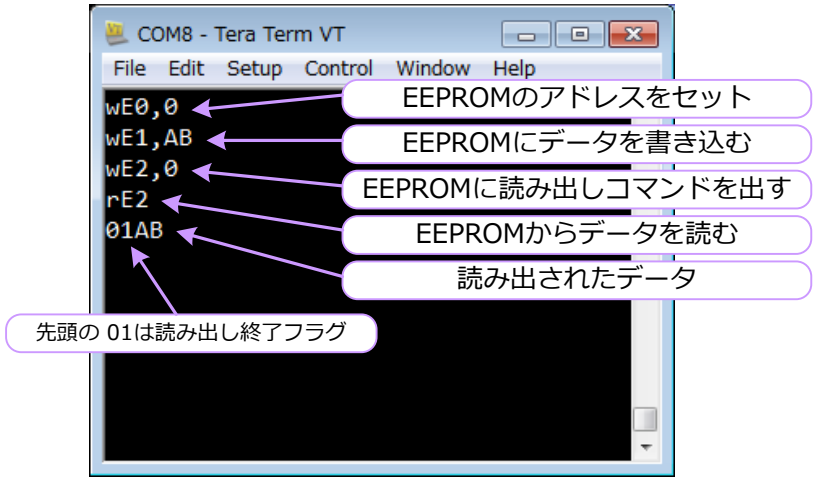
半田ごての先は短いほうが楽です。
長いと先のブレが大きくなります。
電熱線のものより、セラミックが早く(数十秒)、温度も変更できます。
少し高価ですが。

EEPROMにデータを書いたり読み出したりできるモジュールが組み込まれています。

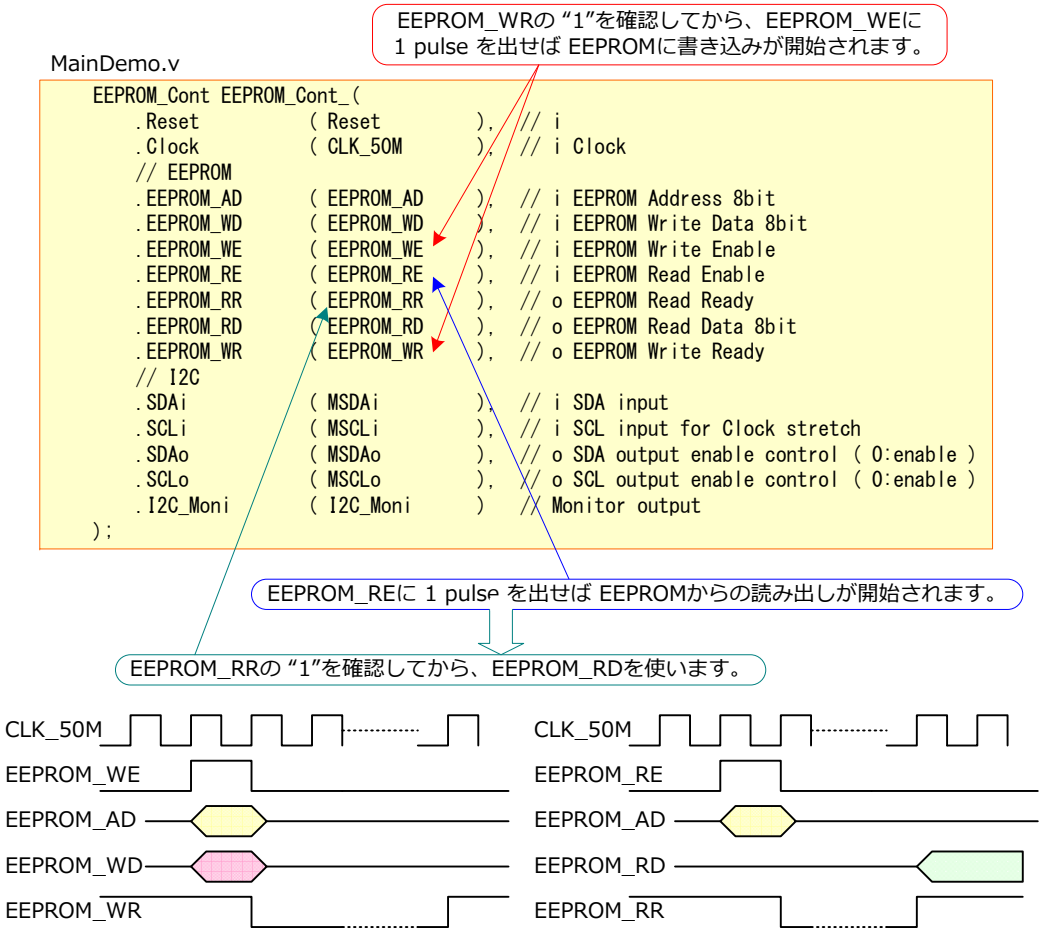


ちょっとした情報を記憶しておくのに便利です。
MAX10には UFMがあるのですが、10M02/08には 1系統しかなく、基板固有情報などは外部にあったほうが便利で、間違って書き変えてしまうミスも少ないと思います。

USBを接続して、ターミナルソフトを立ち上げます。



EEPROMを 新たなモジュールから アクセスするには



iGf02 / iGf08 Demo の USBからアクセスできるレジスタの説明です。

PCのターミナルソフトからアクセスができます。

1. OLEDに表示する数値(Hex)データレジスタ

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D0	TextReg0[15:0]															
D1	TextReg1[15:0]															
D2	TextReg2[15:0]															

Default=0000 , Read/Write

これらのレジスタに数値(Hex)データを書き込むと OLEDモジュールの下段に表示されます。

2. EEPROM テスト用レジスタ

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E0	8'd0								EEPROM_AD[7:0]							

Default=0000 , Read/Write

EEPROMに出力する アドレスをセットします。

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E1	7'd0							_WR	EEPROM_WD[7:0]							

Default=0100 , bit[8]=Read Only bit[7:0]=Read/Write

bit[8] : EEPROM_WR (Write Ready) “1”のとき EEPROMに書き込むことができます。

bit[7:0] : EEPROM_WD (Write Data) EEPROMに書き込むデータです。
書き込みが開始されます。

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E2	7'd0							_RR	EEPROM_RD[7:0]							

Default=0100 , Read Only

bit[8] : EEPROM_RR (Read Ready) “1”のとき EEPROMからデータが読み出しが終了。

bit[7:0] : EEPROM_RD (Read Data) EEPROMから読み出されたデータです。

EEPROMからデータを読み出すには、
① このレジスタに書き込む(データは不定で良い)
② EEPROM_RR が “1”になるまで待つ
③ EEPROM_RR が “1”であるときの bit[7:0] が有効データとなる。

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E3	15'd0															_WC

Default=0001 , Read/Write

bit[0] : EEPROM_WC (Write Control) “1”のとき EEPROMへの書き込み許可。
“0”のとき EEPROMへの書き込み禁止。

3. CTS RTS フロー制御 レジスタ

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E8	11'd0											CTS	3'd0			RTS

Default=0000 , bit[8]=Read Only , bit[0]=Read/Write

bit[8] : USB(UART)からの CTS信号。 (PCでは RTS)
bit[0] : USB(UART)への RTS信号。 (PCでは CTS)

フロー制御をしたいときに使用します。 (Demoでは使っていません)

4. UART Suspend Wakeup レジスタ

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E9	11'd0											SUS	3'd0			WUP

Default=0000 , bit[8]=Read Only , bit[0]=Read/Write

bit[8] : USB(UART)からの Suspend信号。
bit[0] : USB(UART)への Wakeup信号。

5. LED 点灯/消灯 レジスタ

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ED	15'd0															LED

Default=0000 , Read/Write

bit[1:0] : LED “2”のとき LED(D3)が 点滅。
“1”のとき LED(D3)が 点灯。
“0”のとき LED(D3)が 消灯。

6. モニター信号選択レジスタ

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0	MoniSel[31:16]															
F1	MoniSel[15:0]															

Default=0000_0000 , Read/Write

デバッグ用のレジスタで、Demoでは使っていません。

7. I/Oチェック用 レジスタ

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F2	15'd0															CHK

Default=0000 , Read/Write

bit[0] : CheckIO_En “1”のとき 40pinコネクタの IO端子に Hige-Levelの信号がシフトします。
“0”のとき 40pinコネクタの IO端子は全て Loe-Levelになります。

8. バージョン レジスタ

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FF	Version[15:0]															

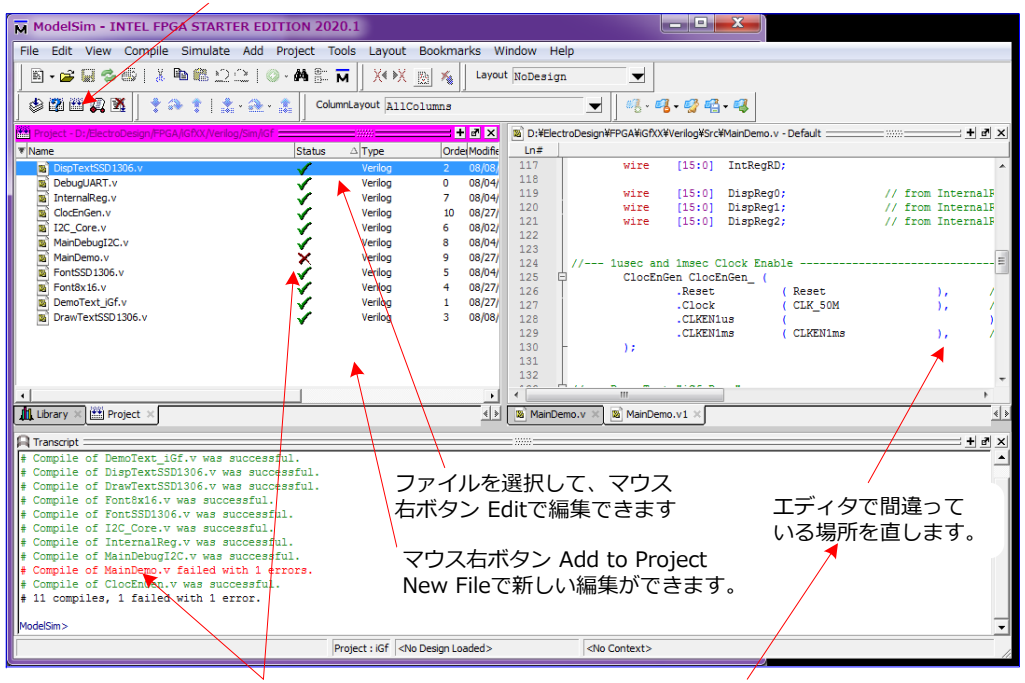
Default=出荷バージョン , Read Only

シミュレータ（ModelSim)と 論理合成 (Quartus Prime)

Intel のダウンロードサイトから、ModelSim, Quartus, MAX10 をダウンロードします。全て無料版でOKです。
使い方は、私が説明するより断然に良いサイトや本がありますので、そちらを参考にしてください。

ModelSimの概要　(Verilog)

このボタンを押すと全てコンパイルしてくれます。



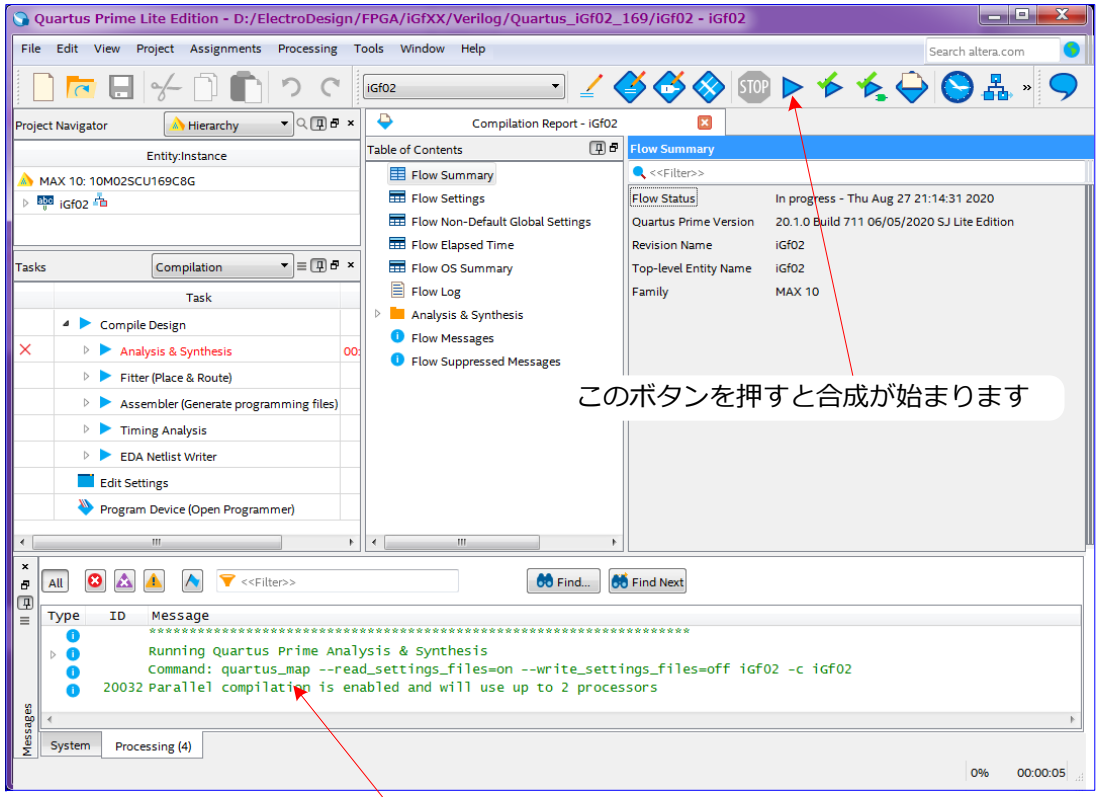
私は WZ Editorを Tab-4で使ってますので、少し文字が合ってません。
この ModelSimで編集もできますので、エディタがなくても始められます。
（日本語のコメントは入れられないようです。）

いきなり、Quartusで論理合成して、エラーが出ると解り難いし、時間がかかります。面倒でも、この ModelSimでチェック、できればシミュレーションするのが、結果的には早いですね。

Editorは 別ウィンドウにしたほうが操作しやすいです。

Quartus Primeの概要

論理合成をして FPGAに書き込むデータを生成するツールです。



ここに エラーが出たら修正。ワーニングは必要に応じて修正です。

次のページのようにすれば、簡単に環境設定できます！

新しくプロジェクトを作るとき、環境設定を最初から作るのは手間がかかります。すでにあるファイルをコピペすると楽です。

Quartus Primeのファイル

このボードを使っている限り、Quartusの環境ファイルを変える必要はありません。ピン番号やその属性など同じだからです。

Quartus_iGf02_169フォルダを全てコピーします。

iGf02.qsfの↓のほうに、各モジュールのファイル名が並んでいます。
ここで 必要に応じて 新たなモジュールに変更、追加をして、
このファイルをダブルクリックすれば、Quartusが立ちあがります。

```
set_global_assignment -name VERILOG_FILE iGf02.v
set_global_assignment -name VERILOG_FILE ../Src/MainDemo.v
set_global_assignment -name VERILOG_FILE ../Src/ClockEnGen.v
set_global_assignment -name VERILOG_FILE ../Src/EEPROM_Cont.v
set_global_assignment -name VERILOG_FILE ../Src/DemoText_iGf.v
set_global_assignment -name VERILOG_FILE ../Src/DispTextSSD1306.v
# set_global_assignment -name VERILOG_FILE ../Src/DrawTextSSD1306.v
set_global_assignment -name VERILOG_FILE ../Src/DrawText_02_SSD1306.v
set_global_assignment -name VERILOG_FILE ../Src/I2C_Core.v
set_global_assignment -name VERILOG_FILE ../Src/DebugUART.v
set_global_assignment -name VERILOG_FILE ../Src/InternalReg.v
set_global_assignment -name VERILOG_FILE ../Src/CheckIO.v
set_global_assignment -name QIP_FILE FontROM/synthesis/FontROM.qip
set_global_assignment -name SDC_FILE iGf02.sdc
```

iGf08ボードの場合は、02を 08と読み変えてください。

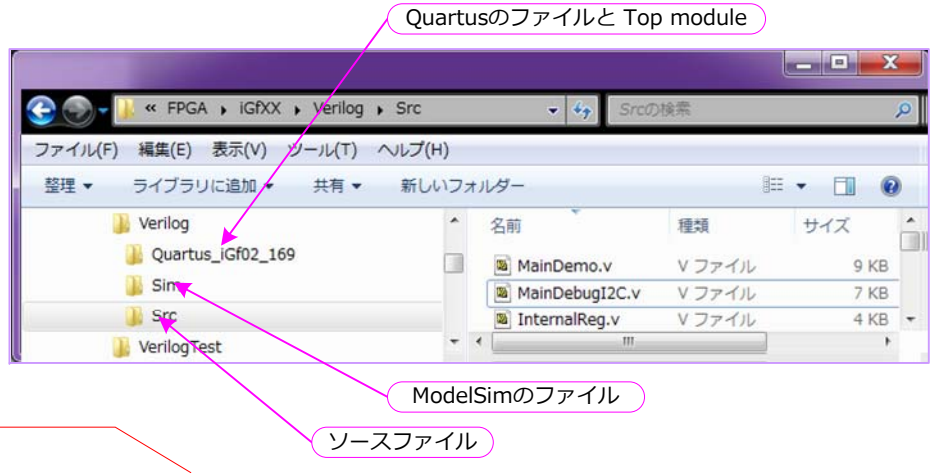
あとは　Quartusの ▶ を押すだけです！

FPGAへの書き込みは次のページで！

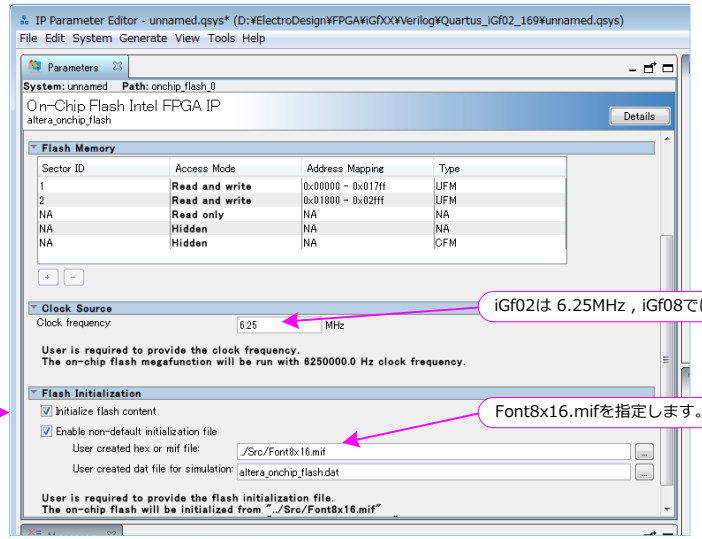
注：ディレクトリ構造が違う場合は **FontROM.qip**を作り直す必要があります。
FPGAのセルを節約するため、Fontの Bit-mapデータは UFMに入っています。
UFMは User Flash Memoryの略で MAX10-FPGA内にあります。

Quartusからこのプロジェクトを読み込むには、
“File” -> “Open Project” -> iGf02.qpf
と選択します。

私のディテクトリの作り方です。（人それぞれでしょうが）



ですから、左の .qsf ファイルは、相対で ../Src/になっています。



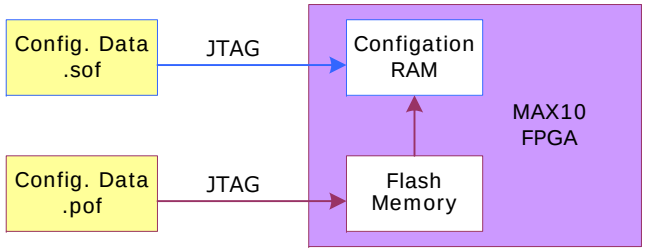
IP Catalog windowが開いてないときは、
View – Utility Windows – IP Catalog で 開く

Libraly
　Basic Functions
　　On Chip Memory
　　On-Chip Flash Intel FPGA IP
　　ここを ダブルクリックで立ち上げる

Entry name は、FontROM – OK とします。

10M02だけ、7.5MHzまでなので
「痛いめに合いました」
その他のデバイスは 116MHz

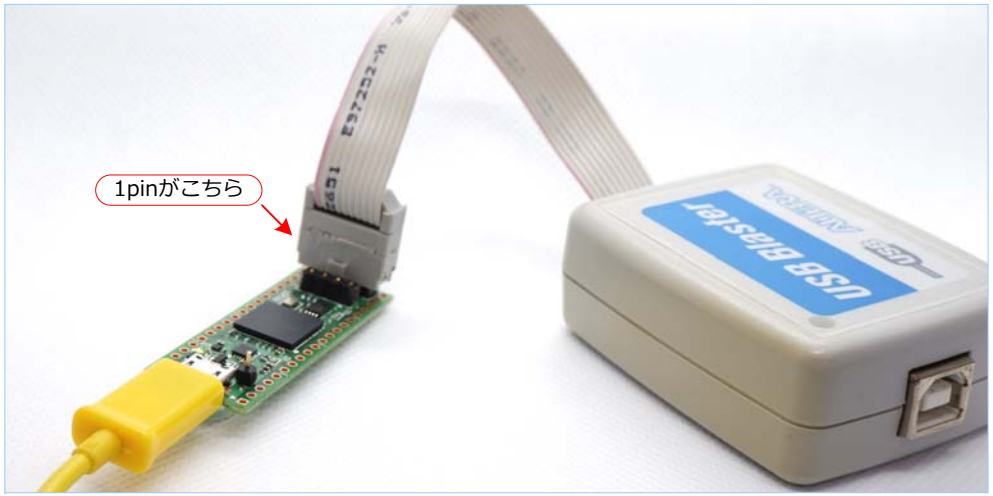
FPGAの書き換えには、Flash Memoryに書き込む方法と
コンフィギュRAMに書く方法の２つがあります。



.sofファイルを指定すると、ロジックセルの動作を決めるメモリに直接書かれます。
書き込みが早いのでデバッグ中は便利ですが、電源を落とすと消えてしまいます。

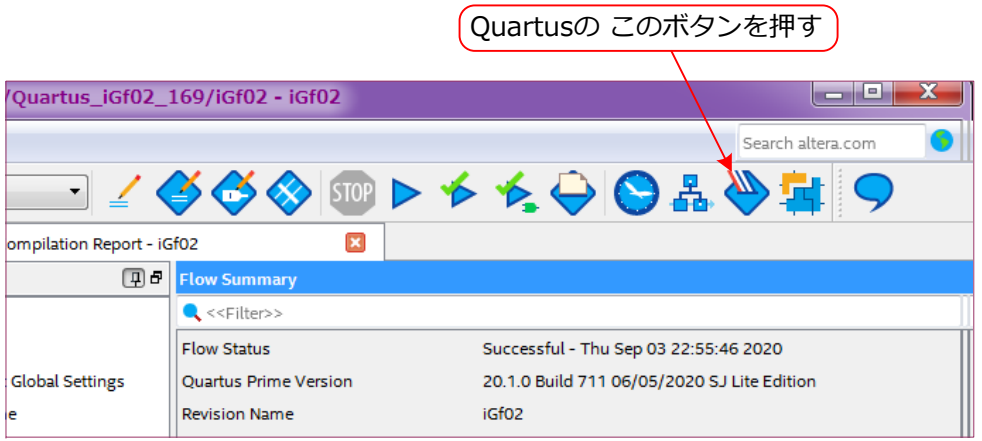
.pofファイルを指定すると、Flash Memoryに書かれます。そして、ロジックセルの動作を決めるメモリに転送されます。書き込みに時間がかかりますが、電源を落としても消えません。
Cyclonデバイスなどは、ファイル変換をしないと Flashに書き込みができませんでしたが、
MAX10では、簡単になりました。

まず、**USB-Blasterを接続**しておく。



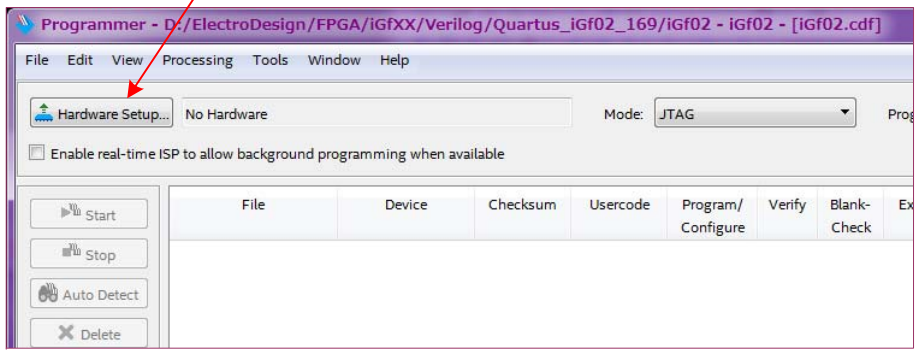
市販されている USB-Blaster

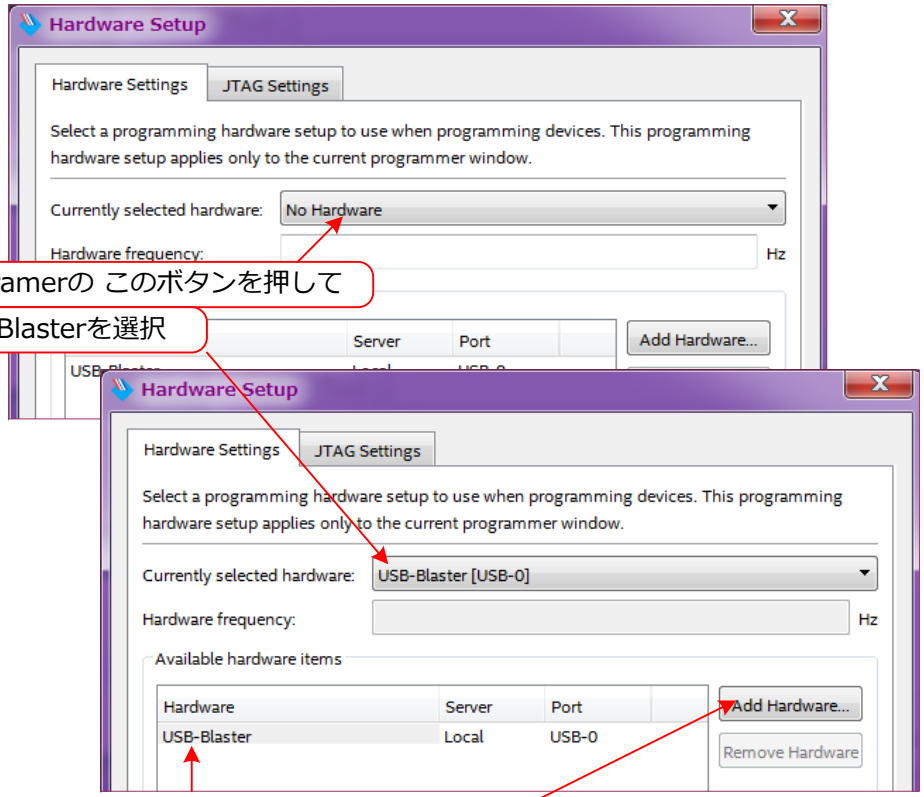
Programerを立ち上げる



USB Blasterが セットアップされてないときは

Programerの このボタンを押す





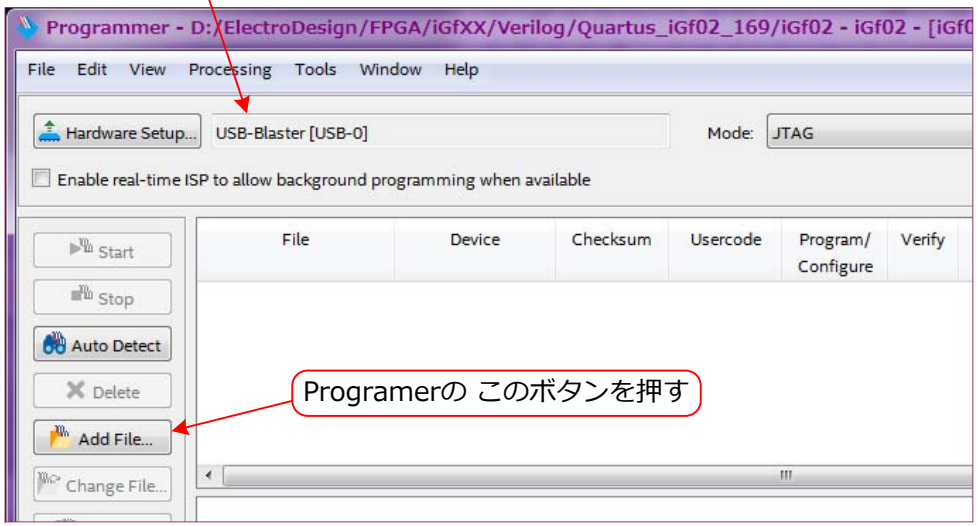
Programerの このボタンを押して

USB-Blasterを選択

ここに表示がないときは、Add Hardware 押して選択

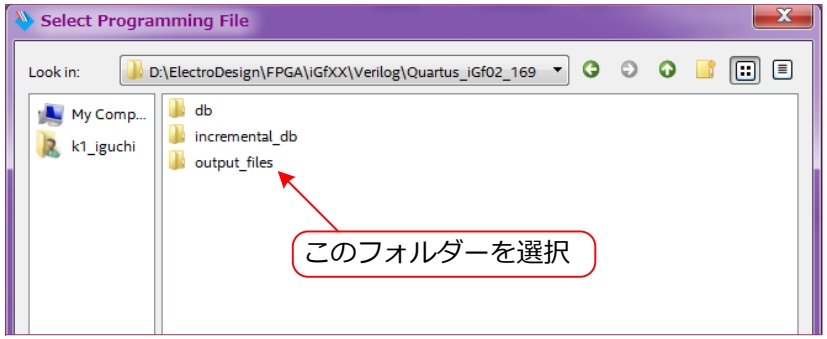
書き込みファイルの選択

USB-Blasterの接続を確認して



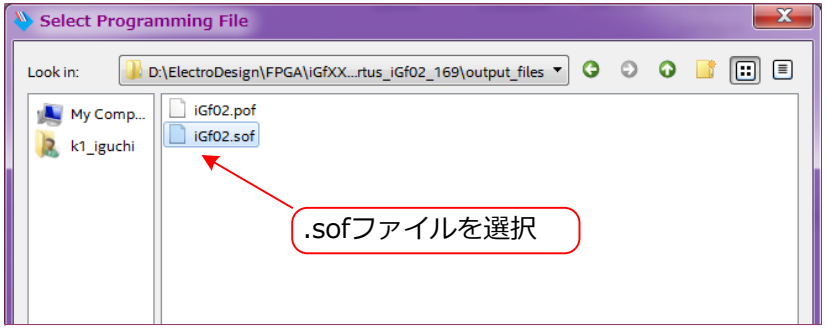
Programerの このボタンを押す

フォルダーの選択

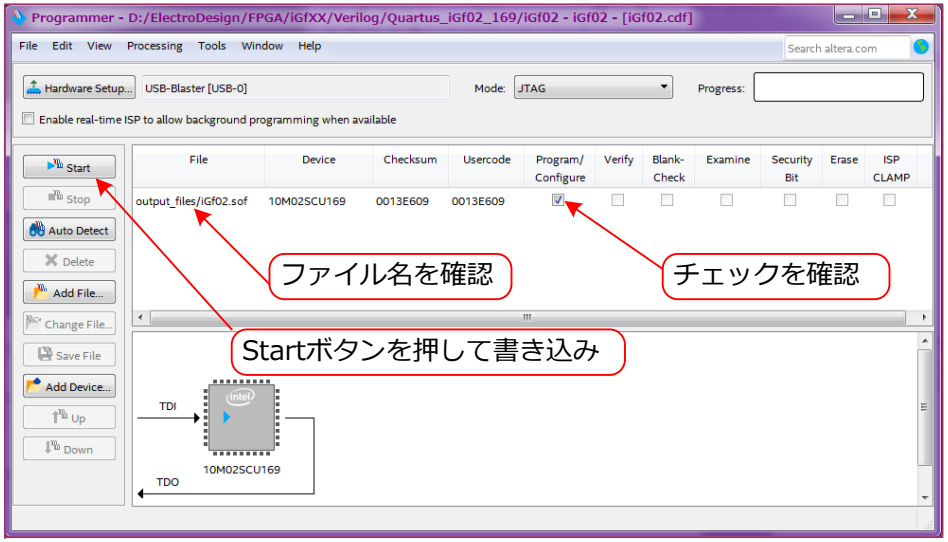


このフォルダーを選択

ファイルの選択

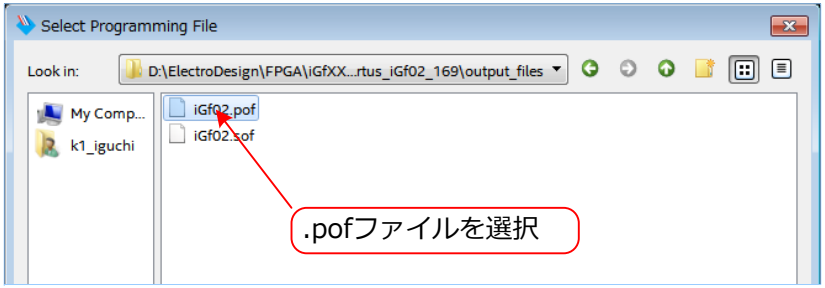


書き込み (.sof)

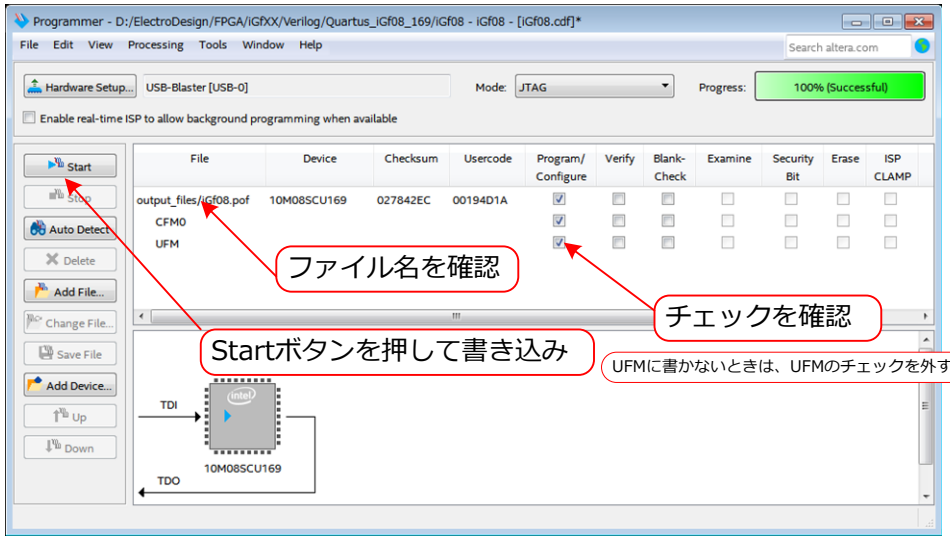


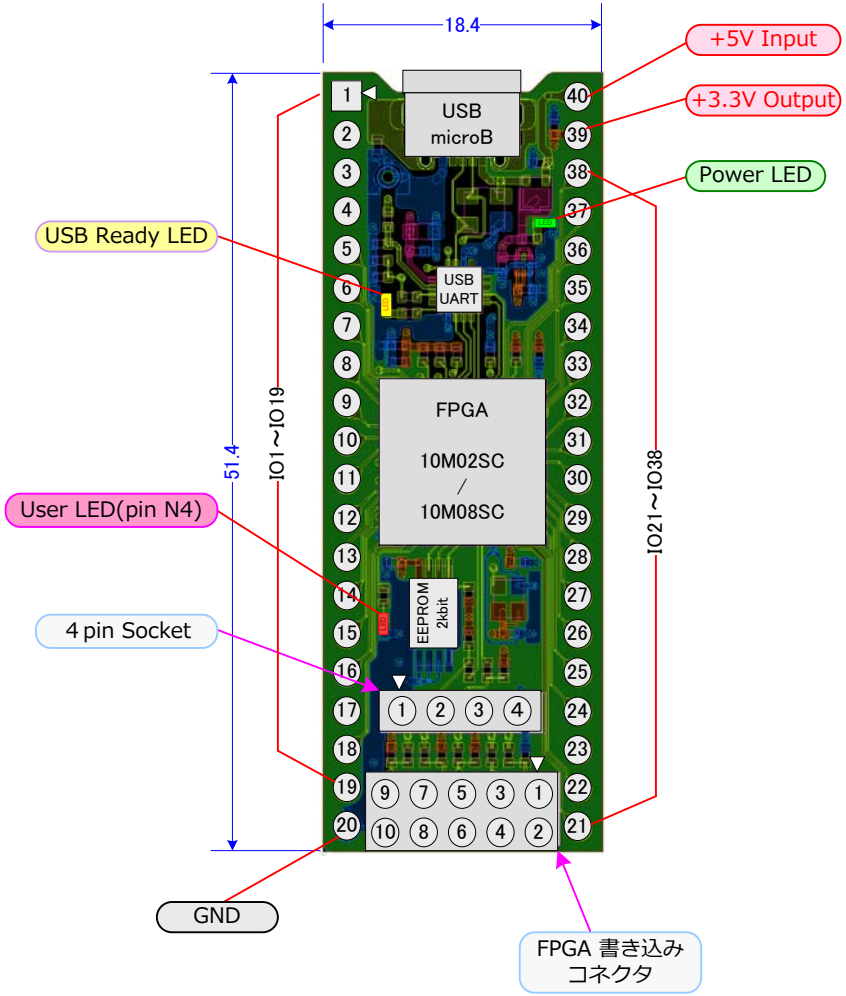
フラッシュ メモリに書き込む場合 (.pof)

ファイルの選択のところで .pof を選択



書き込み (.pof)





注：+3.3V出力の最大電流は *Specifications* のページ参照

J1 USB Connector

Connector Pin No.	Signal Name
1	+5Vi
2	D-
3	D+
4	NC
5	GND

FPGA signals

FPGA Pin No.	Signal Name	IN/OUT
A9	UartTxD	Output
B9	UartRxD	Input
A7	UartRTS	Output
A8	UartCTS	Input
A6	UartWAKEUP	Output
B6	UartSUSPEND	Input

J2 4pin Connector

Connector Pin No.	Signal Name	FPGA Pin No.
1	DSDA	N8
2	DSCL	N9
3	Vcc	
4	GND	
Vcc	Voltage from +5V through diode	

J3 JTAG Connector

Connector Pin No.	Signal Name	FPGA Pin No.
1	TCK	G2
2	GND	
3	TDO	F6
4	+3.3Vo	
5	TMS	G1
6	NC	
7	NC	
8	JTAGEN	E5
9	TDI	F5
10	GND	

J4 40pin Connector

Connector Pin No.	Signal Name	FPGA Pin No.
1	IO_01	A5
2	IO_02	A4
3	IO_03	B4
4	IO_04	A3
5	IO_05	B3
6	IO_06	A2
7	IO_07	B1
8	IO_08	C1
9	IO_09	D1
10	IO_10	E1
11	IO_11	F1
12	IO_12	J1
13	IO_13	K1
14	IO_14	L1
15	IO_15	M1
16	IO_16	N2
17	IO_17	M3
18	IO_18	N3
19	IO_19	H4
20	GND	
21	IO_21	N10
22	IO_22	N11
23	IO_23	M11
24	IO_24	N12
25	IO_25	M13
26	IO_26	L13
27	IO_27	K13
28	IO_28	J13
29	IO_29	H13
30	IO_30	G13
31	IO_31	F13
32	IO_32	E13
33	IO_33	D13
34	IO_34	C13
35	IO_35	B13
36	IO_36	A12
37	IO_37	A11
38	IO_38	A10
39	+3.3Vo	
40	+5Vi	

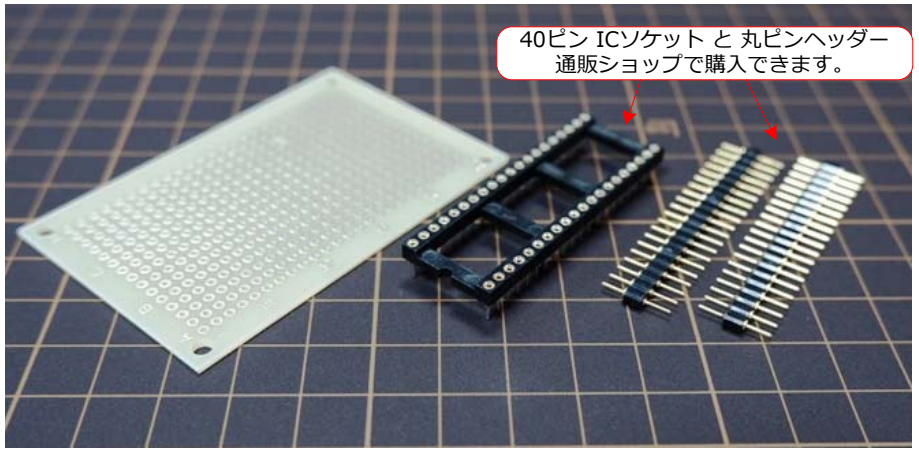
40 pin の コネクタ (20 pin x 2) の取り付けについて

2つの 20pinヘッダーを基板に半田付けする必要があります。

付属のコネクタを半田付けすると、下図のように使うときは便利ですが、基板にソケット（別途購入してください）を付けて差しこむと、高さが高くなってしまいます。そのようなときは、右図のようなソケットを購入すると低く抑えられます。

下図のようなボードは、線や部品を差し込んで使うことができるので、簡単な配線や部品を乗せてテストするときは半田付けしなくて良いので便利です。

実際に組み込むようなものは、コンパクトにしたいし、安定した接続にしたいので、基板上に半田付けして使うのが良いと思います。
ただ、取り外せるようにしたいときは、40pinのICソケットに乗せるのが良いと思います。マイナスドライバーなどで外すときは、基板に傷が付かないように、プラバンを挟むなどしてください。



Electrical Specifications

Item	Symbol	Specifications	Conditions
Supply voltage	Vin	4.5 V to 5.5 V	
Storage temperature	Ts	-20 ℃ to +60 ℃	
Operating temperature	To	0 ℃ to +40 ℃	
Current consumption	Icc	60 mA Typ.	iGf02 Demo To = +25℃
		70 mA Typ.	iGf08 Demo To = +25℃
+3.3V output current	Ioc3	注 1	To = 0 ℃ to +40 ℃
+3.3V output voltage	Vov3	3.3 V ±0.1 V	To = 0 ℃ to +40 ℃
Vcc output current (J2)	Iocc	100 mA Max.	To = 0 ℃ to +40 ℃
Vcc output voltage (J2)	Vovc	Vin – 0.5 V Min.	To = 0 ℃ to +40 ℃

消費電流について

FPGAの消費電力は HDLの内容や動作周波数などによって変化します。

その内容は、“ .pow.summary”ファイルで確認できます。

```
; Power Analyzer Status                ; Successful - Thu Sep 17 09:01:41 2020      ;
; Quartus Prime Version                ; 20.1.0 Build 711 06/05/2020 SJ Lite Edition ;
; Revision Name                        ; iGf08                                       ;
; Top-level Entity Name                ; iGf08                                       ;
; Family                               ; MAX 10                                      ;
; Device                               ; 10M08SCU169C8G                              ;
; Power Models                         ; Final                                       ;
; Total Thermal Power Dissipation      ; 185.51 mW                                   ;
; Core Dynamic Thermal Power Dissipation ; 20.99 mW                                  ;
; Core Static Thermal Power Dissipation  ; 110.84 mW                                 ;
; I/O Thermal Power Dissipation        ; 53.67 mW                                    ;
```

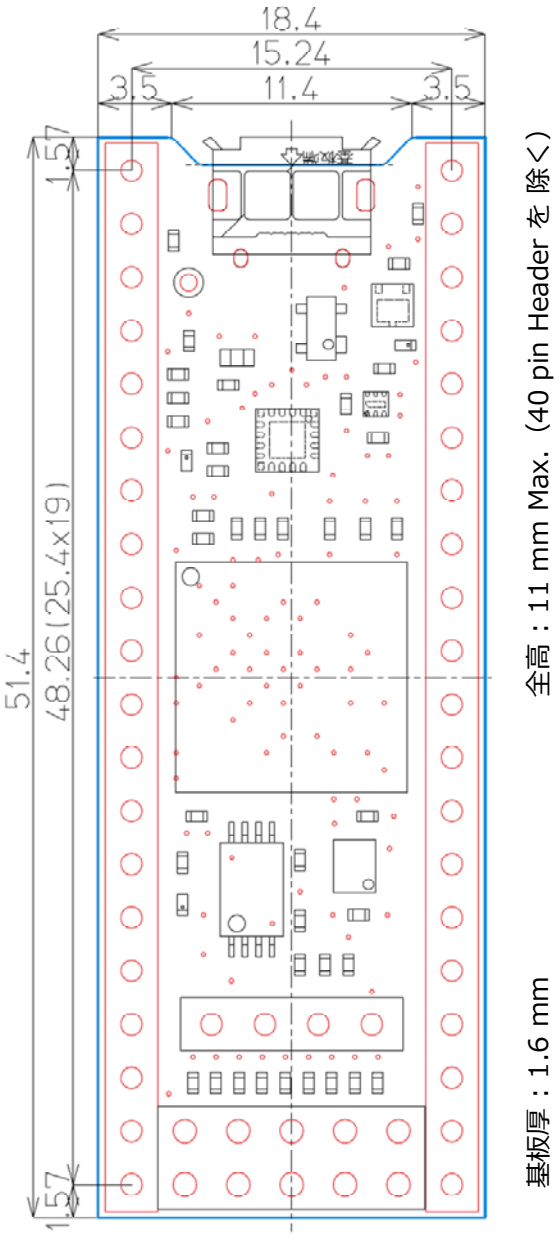
この場合の消費電力は 185.51mW。電源電圧 3.3Vですから、56.2mAとなります。

注１：本ボードで消費する電力は 外部で消費する電力と合わせて 700mW以内に抑えてください。

梱包内容について

Index	Component	Quantity	Comments
1	iGf02 または iGf08	1	FPGA Module
2	20 pin Header	2	FPGA 40 pin 端子用
3	4 pin Header	1	ロープロファイル OLED用
4	OLED Module (128 x 32)	1	32 dot x 128 dot White OLED

Board Dimensions



〒193-0935
東京都八王子市大船町５０４－２
(有)エレクトロデザイン
igf.fpga@gmail.com

日本製