ETCB Manual

HARDWARE & PROGRAMMING MANUAL

株式会社知能機械研究所

ETCB Manual Ver.1.0

http://chinoken.jp
support@chinoken.jp

目次

目次

始めに

始めに

このたびは弊社ロボット制御プログラミングボードETCBをお選びいただきまして誠にありがとうございます。本マニュアルをよくお読みいただいた上でご利用なさいますようお願い申し上げます。

注意事項

- ETCBは電気・電子部品でできています。湿気の多いところやほこりっぽい場所などでのご使用はなさらないでください。同様に濡れた手で扱ったり、電気・電子部品を直接触ったりしてはいけません。
- 仕様に合わない使い方はしないでください。
- 改造をしないでください。
- 動作がおかしいと感じた場合はすぐに使用を止め、すぐに弊社サポートまでご連絡ください。
- Raspberry Piは英国Raspberry Pi財団の登録商標です。

免責事項

ETCBを使用したときのいかなる結果においても弊社では責任を負いません。

同梱品

ETCBには次の部品が付属しています。

- ETCB　1個
- バインド小ねじ M2.6-12　4本
- バインド小ねじ M2.6-4　4本
- 樹脂スペーサΦ2.6L11　4本
- 樹脂スペーサΦ2.6L6　4本
- 2.54ピッチジャンパ　3個
- 2.54ピッチ、13x2ピンソケット　1個

もし足りない場合は下記サポートまでご連絡ください。

サポート

不具合、修理やご質問は下記メールアドレスへご連絡ください。不具合の詳細（不具合の内容、不具合発生時の状況など）とご連絡先（氏名・メールアドレス）を忘れずお書き添えください。内容によりましては弊社より折り返しご質問させていただくことがあります。あらかじめご了解ください。

support@chinoken.jp

ETCB仕様

概要

ETCBは浮動小数点演算回路をもつ最大72MHzで動作するSTM32F302 MCUを搭載したロボット用制御ボードです。ドーターボードとしてRaspberry Piと接続すると、ETCBボードからRaspberry Pi駆動用の電源を供給できます。またUART、I2C、SPIなどの通信ポートを共有してRaspberry Piと通信できます。

外部ポートとしてADC入力ポート、UARTポート、USBポートなどが準備されておりセンサーを使った自律動作するロボットや遠隔操縦ロボットのプログラミングがすぐに始められます。

市販のサーボモーターが使用できるサーボポートを２系統12ポート用意しています。最大6Aの大電流を流せるので大型のロボットも作れます。さらにサーボポートへの電源供給ON/OFFを切り替えることができますので、省電力モードも実現できます。

AtollicのTrueSTUDIO とSTMicroelectornics社のCubeMX、およびST-LINK/v2デバッガに対応しているので、組み込み初心者でも比較的容易にプログラム・デバッグができます。プログラムファイル のUSB書き込み に対応していますので、ST-LINK/v2を持っていなくても書き込みができます。

仕様

項目	サブ項目	内容
電源	推奨入力電圧 最大入力電圧	6.6V～12V 16V
	出力電圧	内部3.3V 外部5.0V
	最大出力電流値 最大連続出力電流値	最大2.2A 2A以下
MCU	型番	STMicroelectronics STM32F302C8
	メモリー	16kB SRAM 64kB Flash
	CLOCK	72MHz（最大・標準）
IO	ADコンバータ	・12bit ADC入力ポートx4（カットオフ周波数1.6kHzのローパスフィルタ付き） ・ADCリファレンス電圧は3.3Vまたは5.0Vのいずれかを選択可能。ただしADC入力電圧は3.3Vを超えてはならない
	サーボ	・シリアルサーボ、コマンドサーボに対応 ・PWMサーボは使用できません ・サーボポート2系統（UART2、UART3）各6ポート ・信号ライン電圧5.0Vサーボに対応 ・UART2、UART3ポートと排他的利用可能 ・サーボ電圧は入力電源電圧と同じ

		・PA15ピンにてサーボへの電源供給ON/OFF切り替え可能 ・サーボポートは1系統あたり最大連続電流値3A以下を推奨
	USB	・PC接続用USBポートx1（FTDI FT230XS経由で仮想COMポートとして使用） ・USBポートx1（MCUのUSB DP、DM端子と接続）
	UART1	・外部拡張ポート（EXT）、COM1ポートと共有可 ・Raspberry Piとポート共有可能
	UART2 UART3	・UART2ポートおよびサーボポートS7～S12に接続、UART2ポートとS1～S6ポートは排他的利用可能 ・UART3ポートおよびサーボポートS1～S6に接続、UART3ポートとS7～S12ポートは排他的利用可能
	I2C	・I2C1として利用 ・Raspberry Piとジャンパピンにて共有可能 ・I2C_SDAおよびI2C_SCL端子は10kΩでプルアップ済み
	SPI	・SPI1として使用 ・Raspberry Piと接続
	GPIO1～GPIO4	・EXTポートに接続 ・MCU GPIOポートまたはTSC_G2_IO1～4ポートとして使用可能
	GPIO5 GPIO6	・Raspberry Pi GPIO22と接続 ・Raspberry Pi GPIO23と接続
LED	D1	・仮想COMポートを使って、MCU側より送信時に点灯（変更可能）
	D2	・EXTポートに接続 ・フルカラーLED ・MCU端子PA8、PA9、PA10にLED RED、GREEN、BLUEがそれぞれ接続 ・PA8～10はAlternate FunctionとしてTIM_CH1～3に変更できるので、PWM出力可能
書き込み	USB1	・DFUポートをショートしてETCBボードを起動すると、PCからDFU形式のプログラムが書き込み可能
	ST-LINK	・EXTポートの3.3V OUT、GND、SWDIO、SWCLK、NRSTポートを使用してSTMicroelectornics ST-LINK/v2でプログラム・デバック可能

対応Raspberry Pi

ETCBが対応しているRaspberry Piは以下の通りです。ただしいずれのRaspberry PiもETCBが供給可能な電流値を超えて使用はできません。

- Raspberry Pi Model B
- Raspberry Pi Model B+
- Raspberry Pi 2 Model B

- Raspberry Pi 3 Model B

注意事項

- 6.6V以下で使用するとETCBの5V出力端子の電圧が低くなります。そのためアナログセンサに5Vを供給する場合は注意してください。またRaspberry Piを取り付けていた場合は画面上に電圧低下警告が表示されることがあります。
- サーボモーターを取り付けている場合は、サーボモーターが動作するのに十分なバッテリーまたは電源を接続してください。電源の性能により供給電力が不十分だったり急激な電圧低下が起こったりするとMCUがリセットされることがあります。
- 一般的にシリアルサーボやコマンドサーボといった、UART（Universal Asynchronous Receiver Transmitter）通信で動作するサーボモーターのみ使用できます。PWMサーボは使用できません。

外部接続詳細

外観・ポート

下図は基板寸法図とポート位置、MCUにピン割り当て、およびEXTポートです。色はポートカラーと合わせています。薄い水色はRaspberry Piの拡張ポートと直接接続されます。

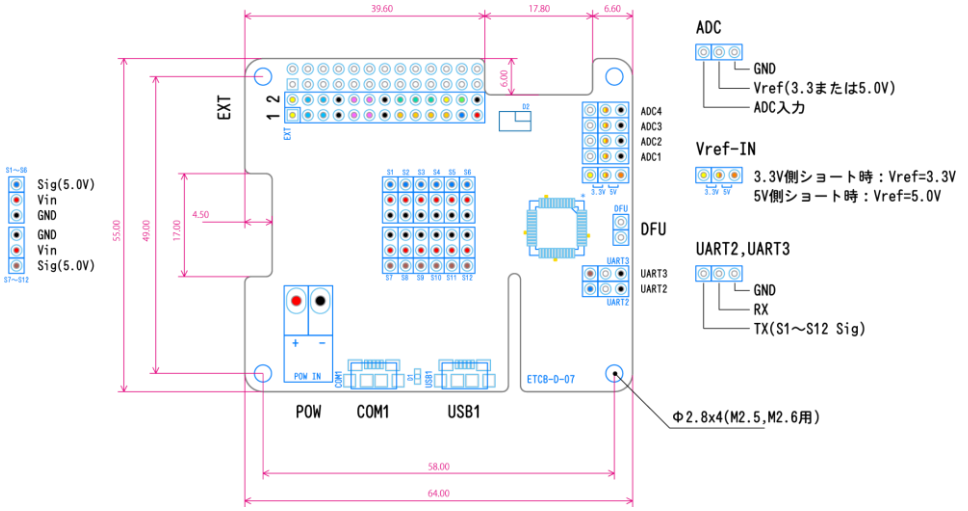


図２　寸法・ポート位置

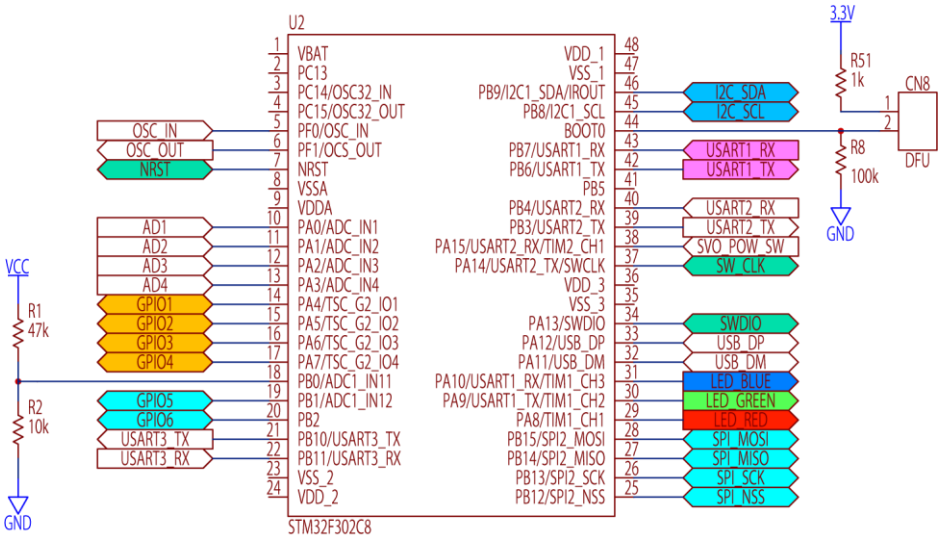


図１　MCUピン配置

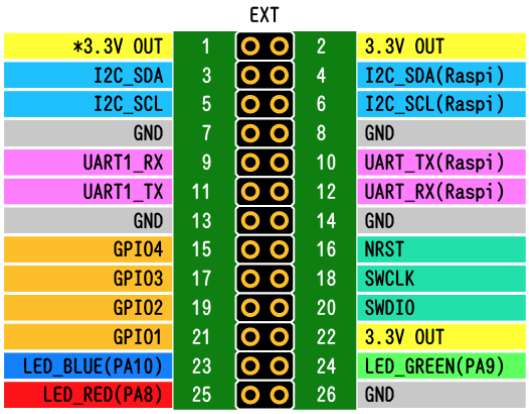


図３　EXTポート

ETCB接続ポート

電源（POW）

電源コネクタは日本圧着端子製造社製のVHコネクタ（S2P-VH）を使っています。＋とーを間違えないで、使用の範囲内の電圧でご利用ください。Li-Fe 2セル、3セルまたはLi-Po2セル、3セルバッテリーが使用できます。

COMポート（COM1）

COM1ポートはMCUのUART1ポートと接続されています。途中にFTDI社USB-シリアル変換ICが入っていますので、PCとUSBケーブルでつなぐとPC側では仮想COMポートとして認識します。FTDI社のデバイスドライバ[1]が必要です。

AD変換ポート（ADC）

ADCポートは12ビットアナログ入力ポートです。切り替えジャンパで3.3Vまたは5.0Vをセンサー側に提供します。

MCUのリファレンス電圧は3.3Vで入力最大電圧も3.3Vになっているので、5.0Vのセンサーをつないだ場合でもADC入力端子の電圧は3.3Vを超えないように注意してください。

---

[1] FTDI社ホームページ（http://www.ftdichip.com/Drivers/VCP.htm）よりVCP（Viratual COM Port）ドライバをダウンロードしてインストールしてください。

UART2・UART3

信号の電圧レベルは3.3Vです。UART2ポートのTX端子はサーボポートS7～S12に接続されています。UART3ポートのTX端子はS1～S6に接続されています。

S1～S2

サーボポートの信号レベルは5.0Vですので市販のサーボモーターが使用できます。MCUポート設定でHalf Duplexに設定してお使いください。UART2、UART3ポートとは排他的使用となります。

EXTポート

UART（UART_TX・UART_RX）

UARTポートはMCUのUART1ポートに接続されています。またCOMポートにも接続されていますので、EXTポートのUARTポートとCOMポートは排他的使用となります。

I2C

MCUのI2C1ポートと接続されています。10kΩでプルアップ済みです。I2Cポートとジャンパピンでショートするとラズベリーパイ4のI2Cポートと接続されます。ショートさせないで外部I2Cデバイスに接続することもできます。外部I2Cデバイスと接続するときはラズベリーパイ4の3.3V端子は使用せず、EXTポート3.3V端子を使用してください。

その他通信ポート

SPI

ラズベリーパイ4のSPIと接続済みです。ラズベリーパイ4とSPI通信するときに使用します。

GPIO1～4

GPIOポートです。

GPIO5・GPIO6

GPIO5とGPIO6ポートはラズベリーパイ4のGPIO2、GPIO3ポートとそれぞれつながっています。ラズベリーパイ4とETCBの両方から出力しないようにしてください。

PWM（フルカラーLED）

LED_RED、LED_GREEN、LED_BLUEはそれぞれMCUのPA8（TIM1_CH1）、PA9（TIM1_CH2）、PA10（TIM1_CH3）とつながっています。フルカラーLEDをPWMで表示する場合は、表示間隔（PWM周期）を10～20msにするときれいに表示されます。

ST-LINK/v2を使うときのETCBとの接続

ST-LINKを使ったプログラム書き込みやデバッグを行う場合は、「ST-LINKを使用する場合」節の図を参考にしてケーブルで接続してください。

開発環境の構築

ハードウェアの設定

ADポート入力電圧の設定

ETCBではADポートリファレンス電圧は3.3Vですが、アナログセンサを駆動できる電圧は3.3Vと5.0Vのいずれか一方を選択できます。お使いのアナログセンサが3.3V系の場合は「外観・ポート」節のVref-IN端子3本の内、左と真ん中の端子を付属のジャンパでショートします。5.0Vの場合は右と真ん中の端子をショートします。どちらもショートしていない場合、ADポートの真ん中の端子はオープンポートとなり、センサーに電源を供給しません。

Raspberry Piとドッキング

ETCBをRaspberry Piとドッキングして使用する場合は付属の13x2ピンソケットをEXTポートの隣の未接続端子に半田付けしてください。

Raspberry Piと通信する場合は、UART通信とI2C通信が使えます。I2C通信をする場合は、I2C_SDA(3)とI2C_SDA(Raspi)(4)を付属のジャンパでショートします。また、I2C_SCL(5)とI2C_SCL(Raspi)(6)もショートします。UART通信をする場合は同様にUART1_RXとUART1_RX(Raspi)をショートし、UART1_TXとUART1_TX(Raspi)もショートします。

UART1端子はUSBのCOM1とも接続していますので、EXTポートのUART1端子をショートすると、ETCBのMCU、Raspberry Pi、USBのTX、RXが全てつながりますが、それらは同時に通信することはできません。MCU－Raspberry Pi間通信またはMCU－USB間通信のどちらかのみ使用してください。

I2C、UARTを単独（MCU単独またはRaspberry Pi単独）で使用したいときは、EXTポートから直接通信ラインを引き出して使用してください。

SPIポート、GPIO5・6ポートはRaspberry Piとハードウェア接続していますので、ジャンパは不要です。

ソフトウェアのインストール

ETCBではプログラム開発のために以下のツールを使用します。

- Atollic TrueSTUDIO Lite：　Atollic社の提供する無料の統合開発環境です。プログラム開発には他にも統合開発環境がありますが、ETCBではTrueSTUDIOを推奨しています。
- DfuSe Demonstration：　USB1端子からプログラムを書き込むためのツールです。ST-LINKを使用する場合は不要です。
- DFU File Manager：　TrueSTUDIOで作成したプログラム（HEX形式）をDFU形式に変換するツールです。DFU形式に変換するとDfuSe DemonstrationツールでUSB端子からプログラムを書き込めるようになります。

- STM32CubeMX： STMicroelectronics社の提供する、C言語コードジェネレーターです。マウス操作で簡単にMCUの初期設定やUSB設定コードを生成できます。コードは、Atollic TrueSTUDIOでインポートして使用できます。生成する内容は、ペリフェラル（周辺機器）のポート設定、クロック周波数、タイマー（周期、PWM設定など）、AD変換（ポート登録、変換時間設定）、UART・I2C・SPIなどの通信設定、DMA転送設定、割り込み設定などほとんどの初期設定項目をマウスで設定し、C言語コードを生成できます。
  設定した項目はプロジェクトファイル（ioc形式）として保存できますので、ETCBではサンプルプログラムを可能な限りSTM32CubeMXのプロジェクトファイルで提供する予定です。

Atollic TrueSTUDIOのインストール

開発環境にはAtollic社のTrueSTUDIOを推奨しています。始めにAtollic社サイトからTrueSTUDIO Lite版をダウンロードしインストールしてください。最新のものを使用するようにしてください。
ダウンロードサイト：http://timor.atollic.com/resources/downloads

インストールの途中でデバッグ用デバイスのドライバインストール画面が表示されますので、ST-LINKにチェックマークを入れてインストールしてください。

DfuSe Demonstrationのインストール

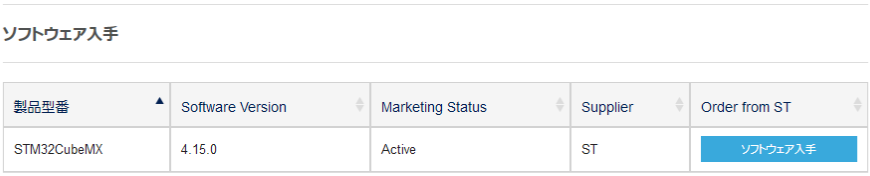
USBポートを使った書き込みをするには、TrueSTUDIOで作成したELF形式実行ファイルをDfuSe Demonstrationソフトウェアを使ってDFU形式(Device Firmware Upgrade)に変更する必要があります。DfuSe Demonstrationソフトウェアは下記手順でダウンロードしてください。ST-LINKを持っている場合は、本ソフトウェアは不要です。

1. STMicroelectronics社サイトへアクセス
   http://www.st.com/content/st_com/ja.html
2. 右上の検索キーワード入力欄で「DFU」または「STSW-STM32080」と入力し、検索します。
3. 検索結果一覧の「製品型番」欄で「STSW-STM32080」を探してください
4. 「STSW-STM32080」がリンクになっていますので、クリックして先へ進みます。
5. 表示されたページの一番下にSTSW-STM32080の「ソフトウェア入手」ボタンがありますので、クリックします。
6. ソフトウェアのライセンス契約内容が表示されますので、よく読んでライセンスに合意する場合はACCEPTボタンをクリックします。
7. 氏名およびメールアドレス入力欄が表示されますので、全て記入し「提出」ボタンをクリックします。
8. ダウンロードリンクが指定したメールアドレスに届きますので、DfuSeソフトウェア（ファイル名：en.stsw-stm32080.zip）をダウンロードしてインストールしてください。

STM32CubeMXのインストール

STM32CubeMXはMCUのポート設定やタイマー、割り込み、DMAなどの諸設定をGUIで行うツールです。ソフトウェア開発用のひな形となるソースコードを出力します。下記の手順でダウンロードしてください。

1. STMicroelectronics社サイトへアクセスします。
   http://www.st.com/content/st_com/ja.html
2. 右上の検索キーワード入力欄で「STM32CubeMX」と入力し、検索します。
3. 検索結果一覧の「製品型番」欄で「STM32CubeMX」を探してください。
4. 「STM32CubeMX」がリンクになっていますので、クリックして先へ進みます。
5. 表示されたページの一番下にSTM32CubeMXの「ソフトウェア入手」ボタンがありますので、クリックします。



6. ソフトウェアのライセンス契約内容が表示されますので、よく読んでライセンスに合意する場合はACCEPTボタンをクリックします。
7. 氏名およびメールアドレス入力欄が表示されますので、全て記入し「提出」ボタンをクリックします。
8. 15番で記入したメールアドレスにダウンロードリンクが届きますので、DfuSeソフトウェア（ファイル名：en.st32cubemx.zip）をダウンロードしてインストールしてください。

STM32CubeMXアップデートとライブラリのアップデート

STM32CubeMXのアップデート方法です。STM32CubeMXでは頻繁に本体やライブラリがアップデートされていますので、常に最新版となるようにしてください。

STM32CubeMXのアップデート

1. STM32CubeMXを起動してください。
2. メインメニューの「Help」メニューから「Check for Updates」を選んでください。
3. 「Check Update Manager」ダイアログが表示されますので、下にある「Check」ボタンで最新版の確認をします。
4. アップデートファイルがあった場合は、「Install Now」ボタンでアップデータのダウンロードとインストール準備を行います。

5. 「Install Now」ボタンでファイルのダウンロードが完了すると、再起動するように指示が出ます。いったんSTM32CubeMXを終了してから、管理者権限で起動してください。STM32CubeMXを管理者権限で起動するには、Windowsショートカットアイコンまたはスタートメニューのアイコンを右クリックして、「管理者として実行」を選択してください。
6. 起動時に自動的にアップデートされます。

ライブラリのアップデート

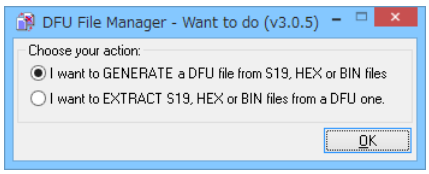
1. STM32CubeMXを起動してください。
2. メインメニューの「Help」メニューから「Check for Updates」を選んでください。
3. メインメニューの「Help」メニューから「Install New Libraries」を選んでください。
4. ダイアログが表示されますので、「STM32CubeF3 Releases」欄から「Firmware Package For STM32 F3」の四角マークをクリックしてチェックマークをいれます。最新のバージョンを選択してください。
5. ダイアログ一番下の「Install Now」ボタンでファームウェアパッケージをインストールします。

プログラム作成と書き込み・実行

DFU形式ファイルをUSBポート経由で書き込み

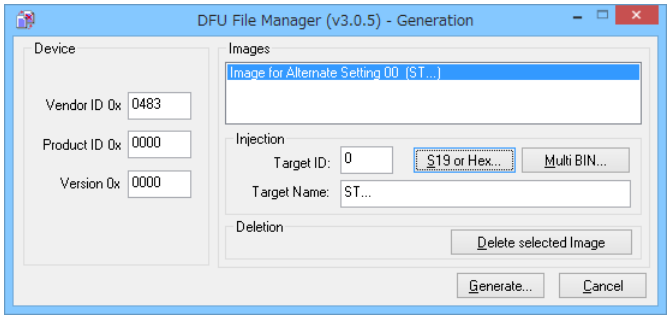
HEXファイルをDFU形式に変換する

1. Dfu file managerを起動します。I want to GENERATE a DFU file from S19, HEX or BIN filesにチェックを入れ、OKボタンを押します。



2. 「S19 or Hex...」ボタンを押して、変換対象のファイルを選択します。今回はサンプルプログラム（LED_TEST.elf.hex）を使用します。

3. Generateボタンを押して、DFUファイルへの変換を実行します。変換後のファイル名の入力が求められますのでDFUファイルの名前（今回はLED_TEST.dfu）を入力して「保存」ボタンを押します。



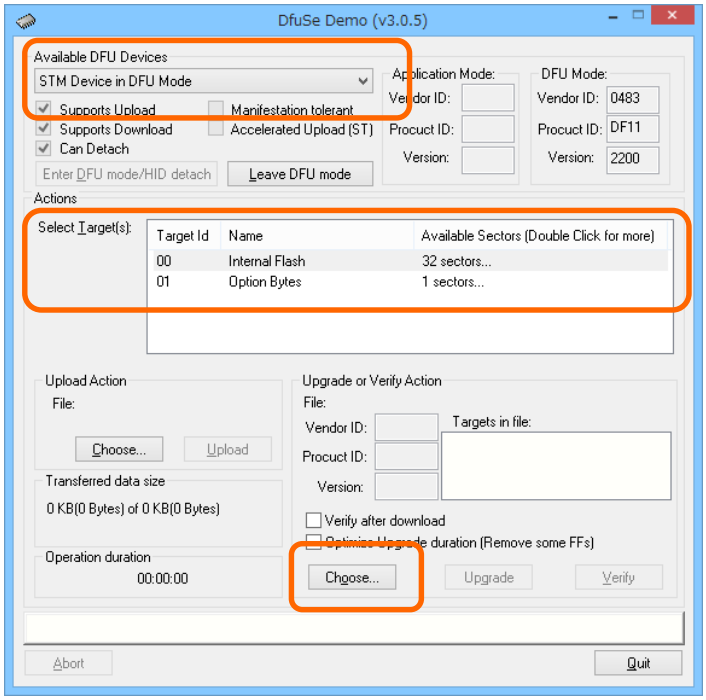
4. ETCBボードのDFU端子をジャンパでショートさせた状態で電源を入れます。ジャンパを刺すためのピンヘッダは付属しておりませんので、ピンヘッダをお客様で半田付けしていただくか、ピンセットのようなものでDFU端子をショートさせてください。半田付けや端子のショートは間違えないように十分気をつけて行うようにしてください。
   USB1のコネクタにマイクロUSBケーブルを取り付けPCと接続します。初回はデバイスドライバ（STM Device in DFU Mode）のインストールが実行されます。

DFUファイルの書き込み

1. DfuSe Demoを起動します。「Available DFU Devices」のプルダウンメニューに「STM Device in DFU Mode」と表示されていることを確認してください。何も表示されていない場合は、4の手順に戻ってDFUモードで起動しなおしてください。
   また、DfuSe Demoウィンドウの真ん中にある、「Select Target(s):」の欄に、「Internal Flash　32sectors...」、「Option bytes　1sectors...」とあるか確認してください。表示が違う場合は、DF

Uモードで起動していません（MCUを認識していない）。この場合も4の手順に戻ってDFUモードで起動しなおしてください。

2. 「Update or Verify Action」の「Choose…」ボタンを押し、転送対象のDFUファイルを選択します。今回は手順3で作成したLED_TEST.dfuを指定します。

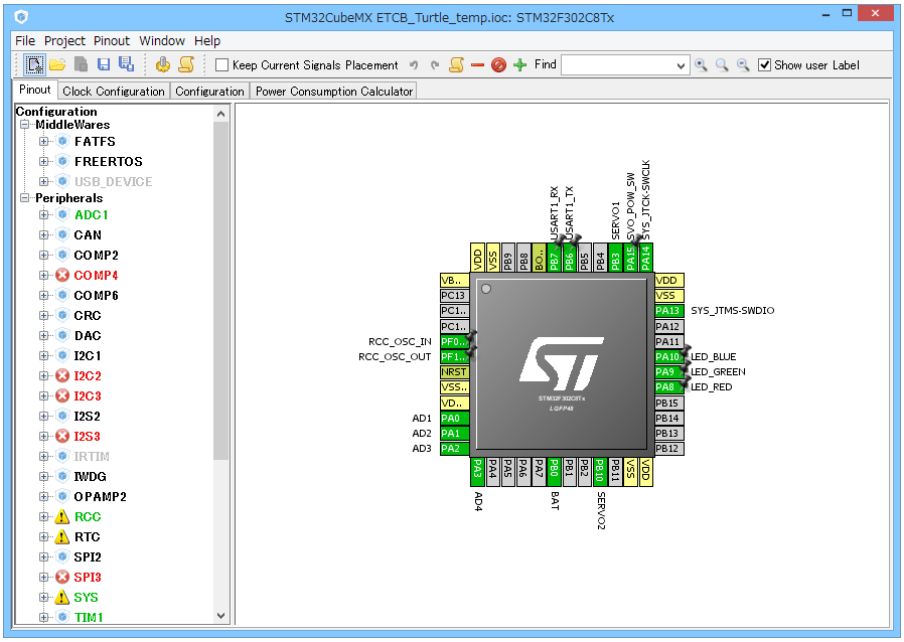


3. Updateボタンを押し、ETCBボードへプログラムを転送します。書き込み中はプログレスバーに進行状況が表示されます。プログレスバーに「Target 00: Upgrade successful !」と表示されたら書き込み完了です。

4. 転送終了後にDFUピンのジャンパを取り外し、電源を入れなおします。

ソースコード編集・新規プログラム作成

STM32CubeMXでソースコード生成



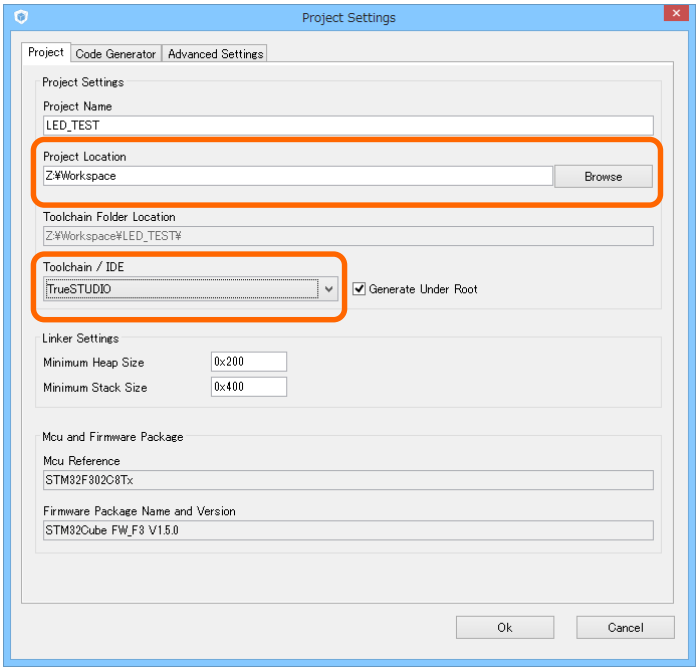
1．ETCB_Turtle_temp.iocをダブルクリックして、「STM32CubeMX」を起動します。
ETCB_Turtle_tempでは、COM1ポートやAD変換は表の構成となっています。

項目	内容
MPU CLOCK	48MHz
ADコンバータ	・AD1、AD2、AD3、AD4、バッテリー電圧、MPU内部温度の6チャンネルのデータを取得 ・変換後のデータはDMAを用いてメモリーへ転送
UART1	・全二重通信 ・通信速度：115200[bps]、データ幅：8ビット、パリティ：無 ・データの送信：DMA転送可、データの受信：割り込み
UART2_TX(PB3), UART3_TX(PB10)	・半二重通信 ・通信速度：115200[bps]、データ幅：8ビット、パリティ：無 ストップビット：1ビット ・割り込み、DMA：未使用 ・双葉産業製RS-304用の設定 ・近藤化学製KRS-2542等を使用する際は、パリティ：EVENにします。
TIM1	・LED_RED、LED_GREEN、LED_BLUE用のPWM信号を生成 ・PWM周期：15ms

	・PWM幅の設定：0～9999
TIM2	・15ms周期のカウンタ ・定期的な処理を組み込む際に使用

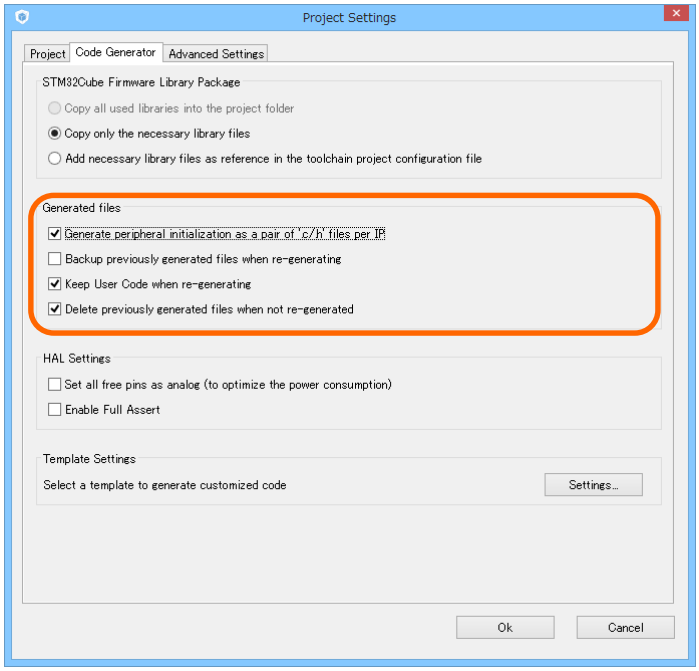
2．Projectメニューから「Settings」を選びます。Project Settingsダイアログが表示されますので、Projectタブを選択してProjectに関する情報を下記のように入力します。「Project Location」の欄は任意です。通常はTrueSTUDIOを初回起動したときに作成されるWorkspaceフォルダを指定します。

また、「Toolchain / IDE」はTrueSTUDIOを選択してください。



項目	内容
Project Name	LED_TEST
Project Location	任意のフォルダ
Toolchain / IDE	TrueSTUDIO
Generate Under Root	チェックを入れます

3．Code Generatorタブに切り替え、「Copy only the necessary library files」を選択します。また「Generated files」欄で「generate peripheral initialization as a pair of 'C/h' fules per IP」にチェックを入れます。その他の欄はデフォルトで問題ありません（下記の図を参考にしてください）。

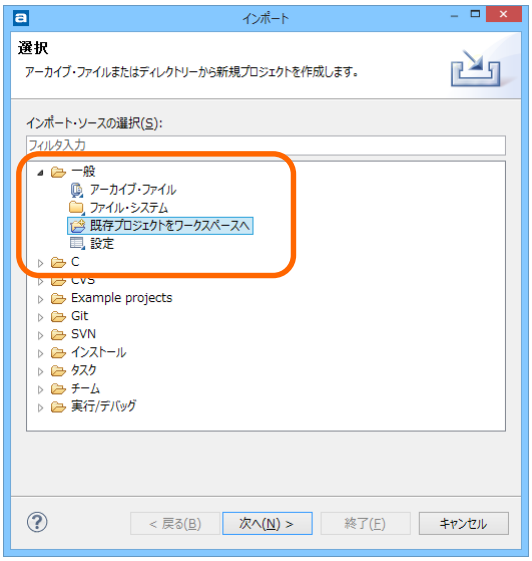


4．Projectメニューから「Generate Code」を選択しソフトウェア開発用のひな形となるソースコードを生成します。ソースコードは２番で指定した、「Toolchain Folder Location」欄に記載されているフォルダに保存されます。

TrueSTUDIOを使ったソースコード編集・デバッグ

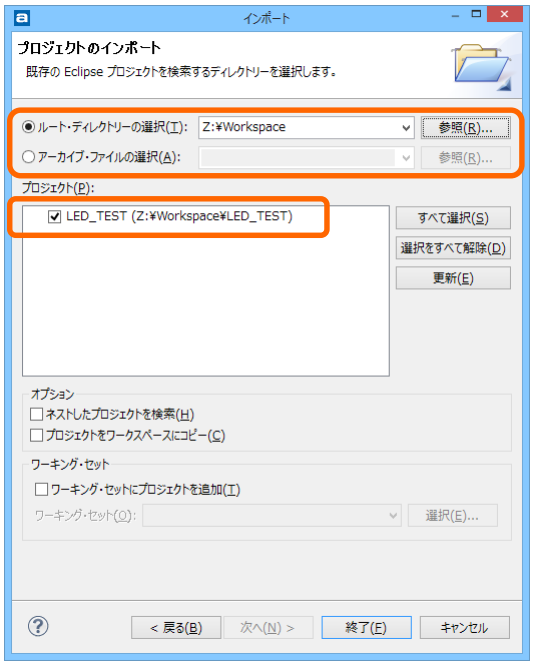
1．TrueSTUDIOを起動します。ワークスペース・ランチャー（ダイアログ）が表示されますので、STM32CubeMXで作成したWorkspaceフォルダを指定してください。

2．「ファイル」メニューから「インポート」を選択します。インポートダイアログ（選択）が表示されますので、「一般」から「既存プロジェクトをワークスペースへ」を選択し、「次へ」ボタンを押します。

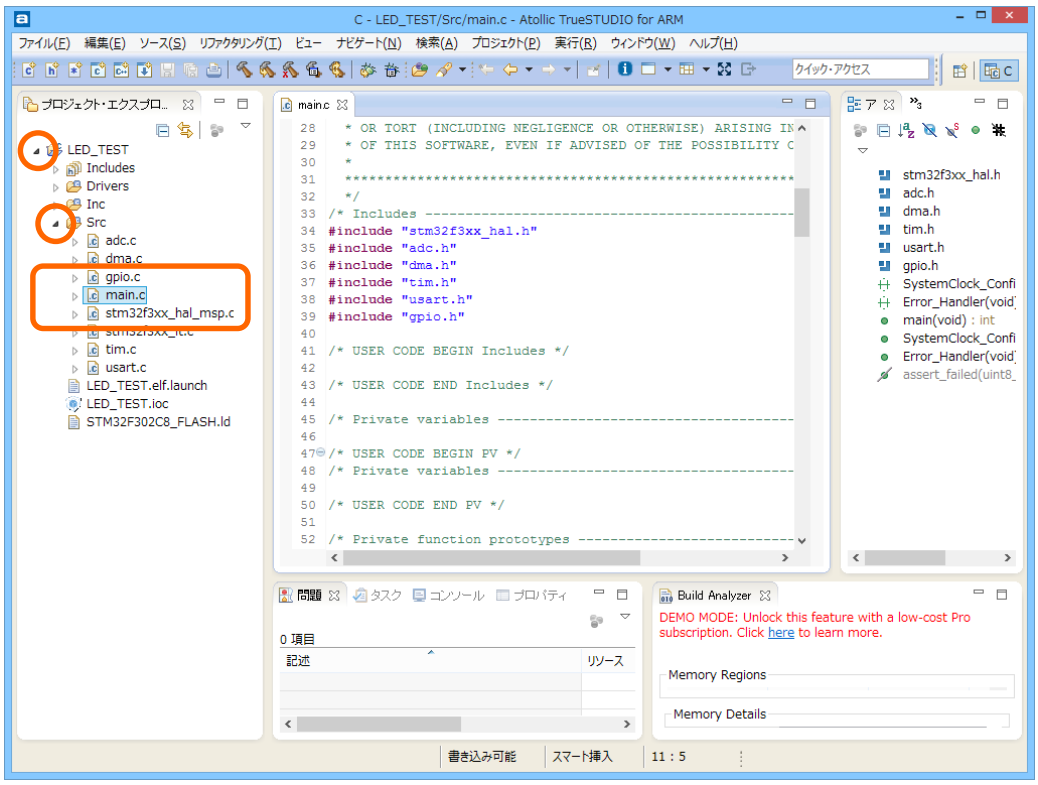


3．インポートダイアログ（プロジェクトのインポート）で「ルート・ディレクトリの選択(T):」を選択し、参照ボタンをクリックしてSTM32CubeMXで作成したWorkspaceフォルダを指定してください。

4．「プロジェクト(P):」欄にWorkspaceにあるLED_TESTプロジェクトが表示されていることを確認して、右下の「終了」ボタンを押します。ボタンを押すとインポートが開始されます。



5．プロジェクト・エクスプローラでLED_TESTの左側の三角マークをクリックしプロジェクトのリストを展開します。Srcの左側の三角マークをクリックしソースファイルのリストを展開します。「main.c」ファイルをダブルクリックし、「main.c」を編集対象とします。



6．main.cにLEDの輝度を調整するコードを追記します（赤字の部分）。なおユーザーが書き入れるコードは必ず「USER CODE BEGIN」から「USER CODE END」の間となります。それ以外の場所にコードを書き入れると、STM32CubeMXでコード生成を行ったときにユーザーコードが消える場合があります。

下記コードをコピー＆ペーストすると、全角スペースが挿入されてしまうことがあります。エラー表示で「stray '@' in program」や「stray '¥201' in program」などと表示された場合は、該当のプログラムに全角スペースが混じっていますので、半角スペースなどに書き換えてください。

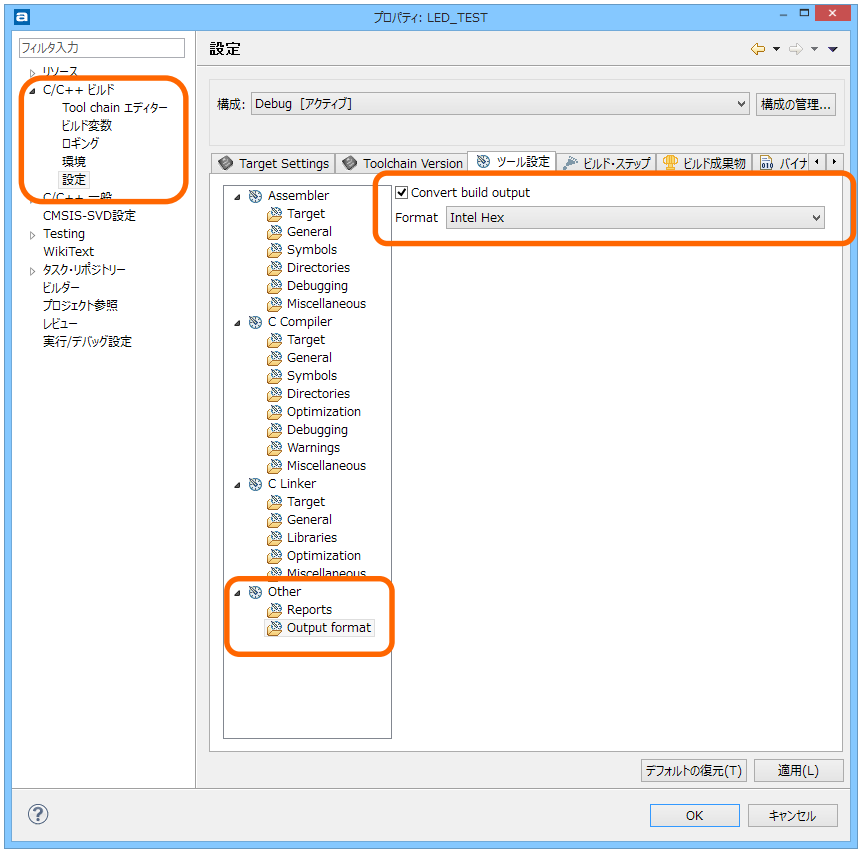
```
/* Private function prototypes -----------------------------------------------*/
void LED(uint8_t , uint8_t , uint8_t);   // LEDのPWMを設定する関数のプロトタイプ

/* USER CODE END PFP */

<中略>

/* USER CODE BEGIN 2 */
```

```
  /*
   * LED用PWM出力開始
   */
    HAL_TIM_Base_Start(&htim1); // TIM1スタート
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); // TIM1_CH1(LED赤) PWM出力開始
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2); // TIM1_CH2(LED緑) PWM出力開始
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3); // TIM1_CH3(LED青) PWM出力開始

    uint8_t count = 0 , red, green;

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
          if( count < 10 )      red = 10*count;         // LED(赤)の設定値
          else                  red = 10*(20-count);
          green = 100 - red;                            // LED(緑)の設定値

          LED(red, green, 0);    // 設定値の反映

          HAL_Delay(100);        // 100ms休む
          count = (count+1) % 20; // 変数countの更新

  }
  /* USER CODE END 3 */

}

<中略>

/* USER CODE BEGIN 4 */
/*
 * RGB明度指定でLEDを点灯する
 * RGB明度は0~100
 */
void LED(uint8_t r, uint8_t g, uint8_t b)
{
        htim1.Instance->CCR1 = (uint32_t)r * 100;
        htim1.Instance->CCR2 = (uint32_t)g * 100;
        htim1.Instance->CCR3 = (uint32_t)b * 100;
}
```

```
/* USER CODE END 4 */
```

7．DFU形式ファイルをUSBポート経由で書き込みを行う場合は、プロジェクトの設定を変更する必要があります。ST-LINKを使用してプログラムの書き込み・デバッグを行う場合はこの手順を省略してかまいません。

8．DFU形式ファイルをUSBポート経由で書き込む場合は、標準の設定で生成されないHexファイルを生成する必要があります。まずTrueSTUDIOのプロジェクトメニューからプロパティを選択し、「C/C++ビルド」の左の三角マークをクリックしてビルド設定の項目を展開します。次に「設定」を選び、「Other」の「Output format」を選択します。「Convert build output」にチェック



を入れ、Formatは「Intel Hex」を選びます。設定が終わったらOKボタンを押してプロジェクトの変更を反映させます。

9．プロジェクトメニューのプロジェクトのビルドを選択し、プログラムのコンパイルを実行します。デバッグ用ELF形式ファイルと一緒にHEXファイルが生成されます。
ビルドに失敗した場合は、プログラムが間違っていないか確認・再編集をして再度ビルドしてください。

書き込みと実行

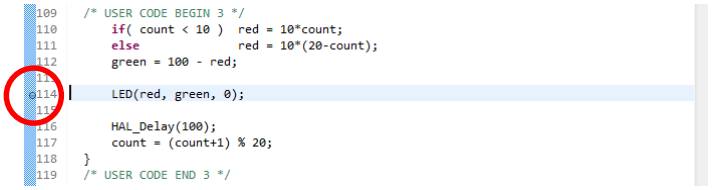
ST-LINKを使用しない場合（USBケーブル書き込み）

「DFU形式ファイルをUSBポート経由で書き込み」で行った手順でビルド後のプログラムをETCBボードへ転送して実行します。

ST-LINKを使用する場合

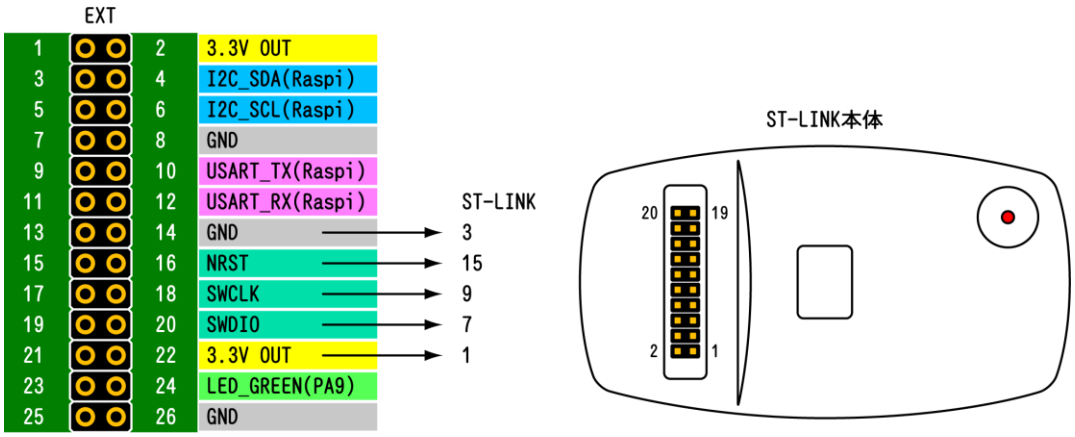
ST-LINKを使用する場合はデバッグを行うことができます。先ほど作成したプログラムにブレイクポイント(デバッグ時に一時停止する)を設定してみましょう。

1．LED(red, green, 0);が記述されている行を選択し、行番号の左側をダブルクリックします。行番号の左側にブレークポイントが設定されていることを示す○マークが表示されます。ブレークポイントを取り除く場合は再度ダブルクリックします。



2．ST-LINKとPCをUSBケーブルで接続し、ETCBボードとST-LINKを接続した状態で、ETCBボードの電源を入れます。EXTポートでのST-LINK差し込み場所を間違えないようにしてください。ST-LINK側とEXTポートのつなぎ方は次のようになります。

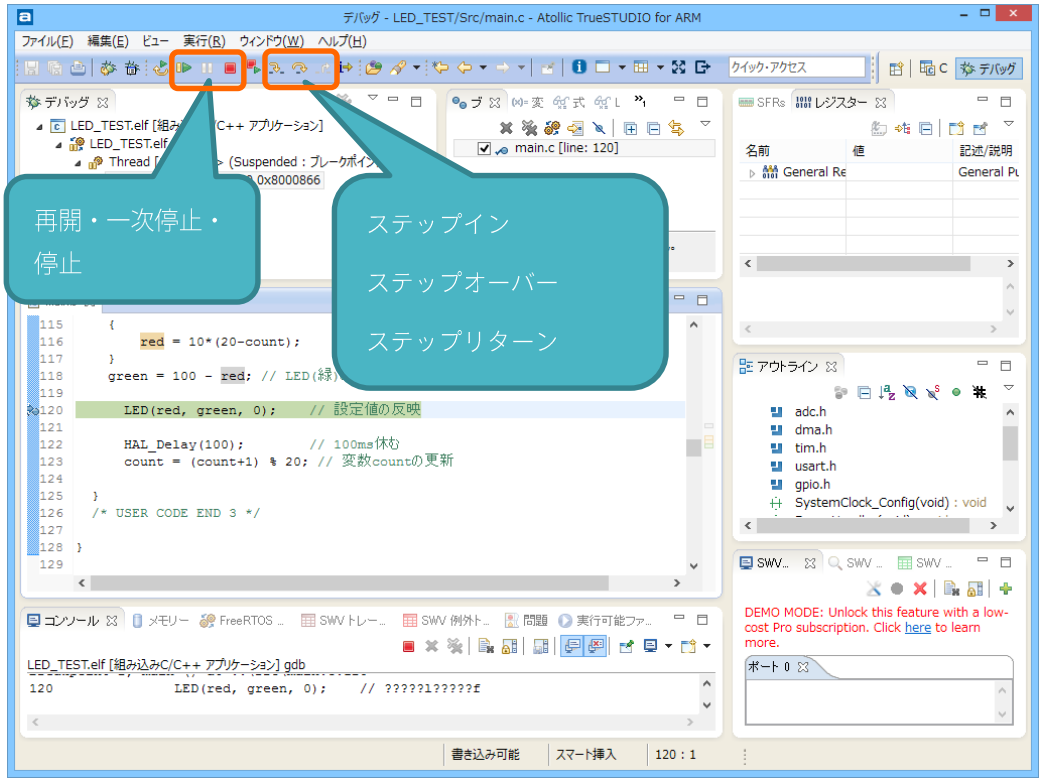
EXTポート番号	SWDIOポート名	ST-LINKコネクタ番号
14	GND	3
16	NRST	15
18	SWCLK	9
20	SWDIO	7



22	3.3V	1

3．実行メニューのデバッグを選択してデバッグモードに移行します。この時ETCBボードのFlashメモリーへ先ほどビルドしたプルグラムが転送されます。

4．実行メニューの再開[2]（F8）を選択します。プログラムの実行が始まり先ほど設定したブレークポイントの行で実行が一時停止します。
実行メニューでステップオーバー[3]（F6）を選択します。一時停止していた処理（LEDの輝度調整）が実行されETCBボード上のLEDが点灯します。



5．プログラムを再開するたびにLEDの色が変化することが確認できます。またデバッグ画面では変数の値を確認・修正することもできます。

[2] 次のブレークポイントまで処理を進めます。ブレークポイントが１つだった場合で、ブレークポイントがループ内に入っていない場合は、プログラムは最後まで実行されます。

[3] 次の命令まで処理を進めます。処理行が関数だった場合は、関数を実行してから次の処理行まで進みます。処理行が関数だった場合で、その関数の中に処理を移したい場合は、ステップイン（F5）を使います。

6．デバッグを終了するときは、実行メニューの終了を選択します。

デバッグ中にブレークポイントで止まらない場合

プログラムの最適化がセットされている場合、デバッグ中にブレークポイントで止まらないで、その前後の行でプログラムが停止することがあります。その場合はTrueSTUDIOのプロジェクトメニューからプロパティを選択し、下図のように「C/C++ビルド」欄の「設定」を開き、「ツール設定」タブから「C Compiler」＞「Optimization」を開き「Optimization Level」を「None（-O0）」にしてください。