マイコン活用シリーズ
ARM/PIC

定番！ARMキット&PIC用

Cプログラムでいきなりマイコン制御

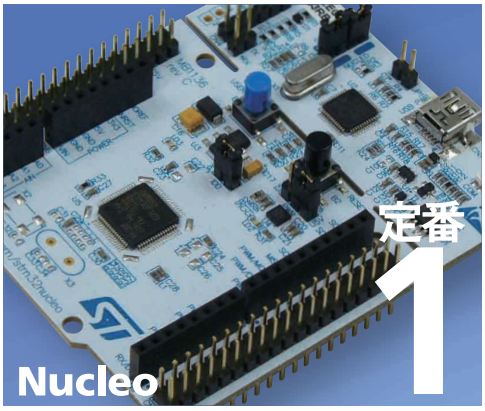
はんだ付け不要！

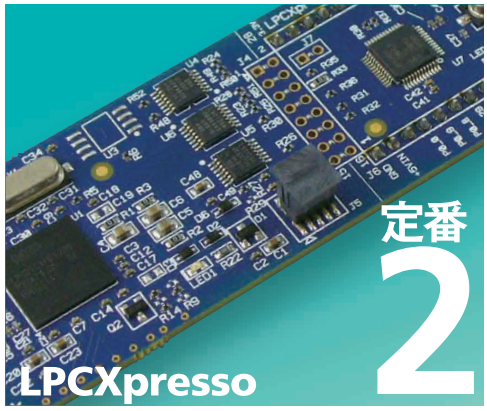
［DVD-ROM付き］

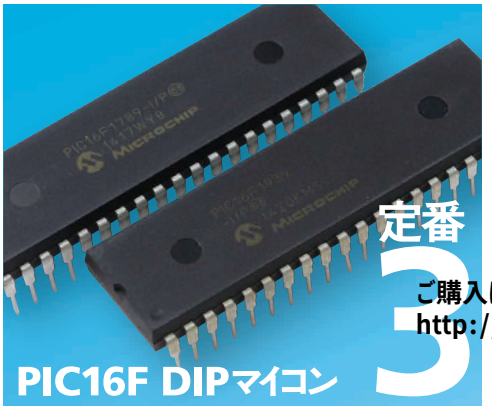
Rapid operations & prototyping with programmed C source code for ready-made ARM/PIC CPU boards

芹井滋喜 著

USB/I²CからA-D/PWMまで，どんな機能もあっさり動かせる







［開発ソフトウェア］
PIC開発環境：MPLAB X IDE＋Cコンパイラ：MPLAB XC8
STM32マイコン用コード・ジェネレータ：STM32CubeMX
LPCマイコン用開発環境：LPCXpresso

［動作確認済みプログラム一式］
GPIO，タイマ，UART，SPI，Microwire，I²C，CAN，PWM，ADC，DAC，コンパレータ，イーサネット，RTC，タッチセンサ，WDT，USBなどの内臓モジュールを使ったプログラム

DVD-ROM付き

ご購入はこちら．
http://shop.cqpub.co.jp/hanbai/books/42/42221.htm

CQ出版社

第1章　開発環境のインストール

使用するマイコン

本書では，比較的よく使われているマイコンの中から，ST マイクロエレクトロニクス（以下，ST 社）の STM32，マイクロチップ・テクノロジー（以下，マイクロチップ社）の PIC16F，そして NXP セミコンダクターズ（以下，NXP 社）の LPC の三つを使用します．

それぞれのマイコンには，同じシリーズでも多種多様のマイコンがありますが，本書で使用するのは次のマイコンとなります．

- STM32
  STM32F103RB《NUCLEO-F103RB》（**写真 1-1**）
  STM32F072RB《NUCLEO-F072RB》
  STM32F401RE《NUCLEO-F401RE》
- PIC16F
  PIC16F1789I/P（40 ピン DIP）（**写真 1-2** 上）
  PIC16F1939I/P（40 ピン DIP）（写真 1-2 下）
- LPC
  LPC1347《LPCXpresso LPC1347》（**写真 1-3**）
  LPC1769《LPCXpresso LPC1769》
  LPC11C24《LPCXpresso LPC11C24》．CAN のテストのみ

LPC と STM32 はマイコン・ボードが市販されているので，マイコン・ボードを MPU トレーナに装着して使用します．上記のマイコン名の後ろの《》内はマイコン・ボードの名称です．PIC16F は DIP IC を直接 MPU トレーナに装着して使用します．

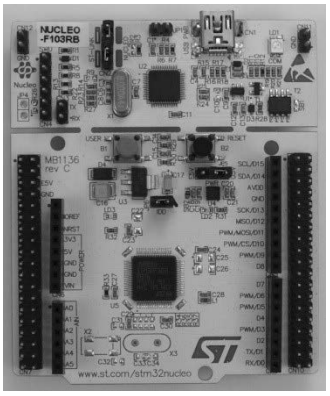


写真 1-1　NUCLEO-F103RB

STM32F072RB，STM32F103RB，STM32F401RE

STM32F072RB，STM32F103RB，STM32F401RE は ST 社のマイコンです．

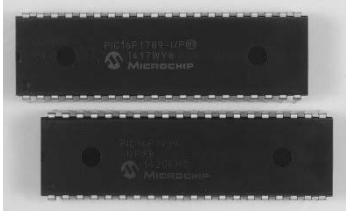


写真 1-2　PIC16F1789/1939

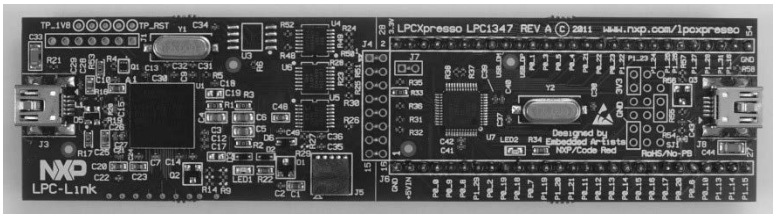


写真 1-3　LPCXpresso LPC1347

表 1-1　本書で使用する STM32（K：KByte）

	CPU	クロック	フラッシュ	SRAM	GPIO	高機能タイマ TIM1
STM32F072RB	Cortex-M0	48MHz	128K	20K	51	1（16bit）
STM32F103RB	Cortex-M3	72MHz	128K	16K	51	1（16bit）
STM32F401RE	Cortex-M4	72MHz	512K	96K	81	1（16bit）

	汎用タイマ	基本タイマ	USART	SPI	I2C	ADC	WDT
STM32F072RB	4（16bit），1（32bit）	2（16bit）	4	2	2	12bit/16ch	1
STM32F103RB	3（16bit）	―	3	2	2	12bit/16ch	2
STM32F401RE	5（16bit），2（32bit）	―	3	3	3	12bit/10ch	2

	USB	CAN	DAC	Comp	Cap	その他
STM32F072RB	1（デバイス）	1	12bit/2ch	2	18ch	CEC，RTC
STM32F103RB	1（デバイス）	1	―	―	―	CRC，RTC，DMA
STM32F401RE	1（OTG FS）	―	―	―	―	SDIO，RTC，DMA

CEC：Consumer Electronics Control，CRC：Cyclic Redundancy Check
RTC：Real-Time Clock，DMA：Direct Memory Access，SDIO：Secure Digital Input/Output

表 1-2　本書で使用する PIC16F（W：Word，B：Byte）

	CPU	クロック	フラッシュ	SRAM	EEPROM	GPIO	汎用タイマ
PIC16F1789I/P	PIC16	32MHz	16384W	1024B	256B	36	2（8bit），1（16bit）
PIC16F1939I/P	PIC16	32MHz	16384W	2048B	256B	36	4（8bit），1（16bit）

	PSMC	ECCP	CCP	UART	MSSP（I^2C/SPI）	ADC	DAC
PIC16F1789I/P	4	―	3	1	1	12bit/15ch	1（8bit），3（5bit）
PIC16F1939I/P	―	3	2	1	2	10bit/14ch	―

	Comp	OPA	Cap Sense	LCD
PIC16F1789I/P	4	3	―	―
PIC16F1939I/P	2	―	16ch	24seg

PSMC：Programmable Switch Mode Control
CCP：Capture/Compare/PWM，ECCP：Enhanced CCP
MSSP：Master Synchronous Serial Port，Comp：Comparator，OPA：Operational Amplifier

コア CPU は，3 製品とも ARM 社の Cortex シリーズで，STM32F072RB が Cortex-M0，STM32F103RB が Cortex-M3，STM32F401RE が Cortex-M4 となっています（**表 1-1**）．

STM32F103RB は Cortex-M3 内蔵マイコンとしては比較的初期の製品です．ARM の Cortex-M シリーズは，おおざっぱに言うと数字が大きいほど高性能で，数字が小さいほどロー・コストと考えてよいでしょう．

PIC16F1789，PIC16F1939

PIC16F1789 と PIC16F1939 はマイクロチップ社のマイコン製品の一つです．このデバイスはミッドレンジと呼ばれるシリーズの一製品で，比較的ポピュラな製品です．PIC16F は多くのデバイスがあり

表 1-3　本書で使用する LPC（K：KByte）

	CPU	クロック	フラッシュ	SRAM	EEPROM	GPIO	汎用タイマ
LPC11C24	Cortex-M0	50MHz	32K	8K	―	36	2（16bit），2（32bit）
LPC1347	Cortex-M3	72MHz	64K	8K+2K+2K	4K	40	2（16bit），2（32bit）
LPC1769	Cortex-M3	120MHz	512K	64K	―	70	4（32bit）

	UART	SPI	SSP	I^2C	ADC	DAC	WDT
LPC11C24	1	2	―	1	10bit/8ch	―	1
LPC1347	1	―	2	1	12bit/8ch	―	1
LPC1769	4	2	2	3	12bit/8ch	10bit	1

	USB	イーサネット	CAN	その他
LPC11C24	―	―	1（トランシーバ付き）	
LPC1347	1（デバイス）	―	―	
LPC1769	1（デバイス/ホスト/OTG）	1	2	RTC，I^2S，PWM など

ますが PIC16F シリーズであれば使い方はほぼ同じで，内蔵モジュールやピン数の組み合わせによって製品が分かれていると考えてよいでしょう（**表 1-2**）．

LPC1347，LPC1769，LPC11C24

LPC1347，LPC1769，LPC11C24 は NXP 社のマイコンです．LPC11C24 は Cortex-M0，LPC1347 と LPC1769 は Cortex-M3 をコアに持っています（**表 1-3**）．

LPC1769 はイーサネット，リアルタイム・クロックのテストに使います．LPC11C24 は CAN のテストにだけ使います．

ARM Cortex マイコンの互換性

ARM アーキテクチャのマイコンはいろいろなメーカから発売されています．同様に開発環境も ARM 社以外からも発売されているので，これらを利用すれば開発環境を共通化することができます．ただし，ARM アーキテクチャで共通化されるのはコア CPU のみです．

昨今のマイコンにはさまざまなモジュールが内蔵されていますが，これらの内蔵モジュールはメーカごとに独自のものを搭載しているので，同じ ARM コアを持つマイコンであってもメーカが異なれば別のマイコンと見た方がよいでしょう．

マイコンを選択する際には，どのような内蔵モジュールが使用できるかが重要なポイントとなる場合が多いのですが，選択したマイコンが今まで使っていたメーカとは異なっていた場合は同じ ARM アーキテクチャであっても内蔵モジュールの使い方は共通ではないので注意が必要です．

使用 OS と開発環境

本書では Windows 7，Windows 8.1，Windows 10 を前提に解説を進めます．ほかの OS では動作確認は行っていません．

STM32 の開発環境

STM32 の開発環境は以下を使用します.

- 統合開発環境，C コンパイラ　MDK-ARM（MDK-Lite） Ver.5.17（ARM 社）
- コンフィグレーション・ツール　STM32CubeMX Ver.4.11.0（ST 社）
- ライブラリ　HAL（ST 社）
- インサーキット・デバッガ　ST-LINK（Nucleo ボードに搭載済み）

統合開発環境，C コンパイラには ARM 社の MDK-ARM の MDK-Lite 版を使用します．このツールは，統合開発環境 µVision と C コンパイラ ARM コンパイラがいっしょになっているので，MDK-ARM のみでプログラム開発を行うことができます.

STM32 の Nucleo ボードは，インサーキット・デバッガ ST-LINK の回路を搭載しているので，書き込み/デバッグ用アダプタのようなものは必要ありません.

STM32CubeMX は，ST 社のコンフィグレーション・ツールです．PIC16F には Code Configurator という同じような機能の製品があります. このソフトウェアは無料で, ARM 社の MDK-ARM のほか IAR 社の EWARM や GCC でも使用することができます.

STM32CubeMX は HAL（Hardware Abstraction Layer）と呼ばれる ST 社のライブラリを使用するコードを生成します．HAL は，マイコンごとに用意されていますが，使い方は統一されています．HAL は STM32CubMX でコンフィグレーションを行う際, 必要なライブラリが自動でダウンロードされるので個別にダウンロードする必要はありません.

PIC16F の開発環境

PIC16F の開発環境は以下を使用します.

- 統合開発環境　MPLAB X IDE Ver.3.15
- C コンパイラ　XC8 Ver.1.35
- コンフィグレーション用ツール　MPLAB Code Configurator Ver.2.52.2
- インサーキット・デバッガ　PICkit3

統合開発環境は，マイクロチップ社の MPLAB X IDE を使用します． MPLAB X IDE は，マイクロチップ社のウェブ・ページから無償でダウンロードできます.

C コンパイラの XC8 はマイクロチップ社の C コンパイラで, 有償版と機能限定の無償版があります.

MPLAB Code Configurator は，PIC16F を使う際の初期設定用のコードを生成するツールです．このツールがなくてもソフトウェアの開発を行うことができますが，マルチファンクションのピンの設定など，面倒でまちがえやすいコードを視覚的にピンを確認しながらソース・コードを自動生成することができるので，かなり便利なツールになっています.

写真 1-4　インサーキット・デバッガ PICkit3

プログラムの書き込みはマイクロチップ社のインサーキット・デバッガ PICkit3（**写真 1-4**）を使用します．プログラムのダウンロードだけではなく，デバッグ機能に対応した PIC16F であればデバッガとしても利用することができます．

LPC の開発環境

LPC の開発環境は以下を使用します．

- 統合開発環境　　LPCXpresso IDE Ver.7.9.2（NXP 社）
- ライブラリ　　LPCOpen Ver.2.05（NXP 社）
- インサーキット・デバッガ　　LPC-Link（LPCXpresso ボードに搭載済み）

LPCXpresso IDE は NXP 社の統合開発環境で，有償版のほか無償版があります．ここでは無償版を使用します．この開発環境には C コンパイラ（GCC）が含まれています．また，ライブラリの LPCOpen も LPCXpresso IDE と同時にインストールされます．

LPCXpresso ボードには，インサーキット・デバッガ LPC-Link の回路が搭載されているので，ボードのほかには LPCXpresso IDE だけをインストールすれば，すぐに開発を始めることができます．

開発環境のインストール

開発を始めるに当たって，最初に開発環境のインストールを行います．ここでは，それぞれのマイコンの開発環境のインストール方法を簡単に説明します．

開発環境はマイコンごとに異なったツールを使用しており共通して使用するツールはありません．

STM32 開発環境のインストール

STM32 は ARM Cortex-M シリーズをコアに持つマイコンで，多くの ARM 系のコンパイラでサポートされていますが，ここでは ARM 社の MDK-ARM を使用します．

MDK-ARM は機能が豊富で使いやすい開発環境です．このコンパイラには 32KByte 制限の無償版（MDK-Lite 版）と，STM32L0/F0 シリーズ用のプログラム・サイズの制限のない無償版があります．STM32L0/F0 シリーズだけを利用する場合はこちらの版の方が便利です．

MDK-ARM の無償版は，それぞれ次のリンクからダウンロードできます．

- MDK-ARM 32KByte 制限の MDK-Lite 版
  https://www.keil.com/demo/eval/arm.htm
- MDK-ARM STM32L0/F0 専用無償版
  http://www2.keil.com/stmicroelectronics-stm32/mdk

このサイトで必要事項を記入して［Submit］ボタンを押すと，MDK-ARM のダウンロード画面になります．

画面下の MDK517.EXE をクリックしてプログラムをダウンロードし MDK517.EXE を実行しインストールを行います．MDK-ARM のインストール画面は付属 DVD-ROM を参照してください．

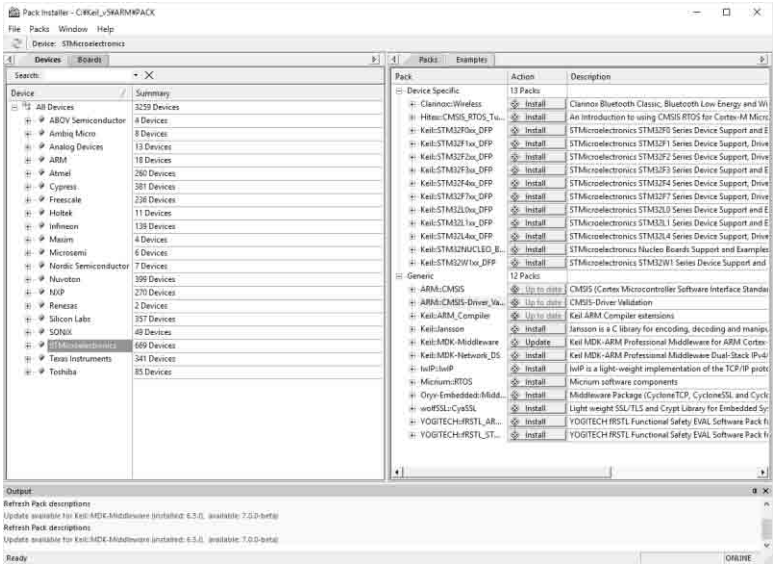


図 1-1　Pack Installer の画面

MDK-ARM のインストールが終わると，図 1-1 のように，Pack Installer が起動します．MDK-ARM はさまざまなマイコンに対応していますが，マイコンのサポート・ファイルは，この Pack Installer で必要なものをインストールするようになっています．

STM32 のサポート・ファイルをインストールするには，まず左側のウィンドウで図のように「STMicroelectronics」を選択し，「File」メニューから「Refresh」を選択すると画面右側のウィンドウが最新の情報に更新されます．そして，右の画面から必要なファイルを［Install］ボタンでインストールします．

ここでは，次のパッケージをインストールしておきます．

Keil::STM32F0xx_DFP
Keil::STM32F1xx_DFP
Keil::STM32F4xx_DFP
Keil::STM32Nucleo_BSP

STM32CubeMX のインストール

MDK-ARM のインストールが終わったら STM32CubeMX をインストールします（図 1-2）．STM32CubeMX は，次のリンクからダウンロードできます．付属 DVD-ROM にも収録されています．

http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1533/PF259242?sc=stm32cube

STM32CubeMX のインストール画面は付属 DVD-ROM を参照してください．

STM32CubeMX のインストールには，Java ランタイム（JRE）のインストールが必要になります．Java がインストールされていない場合は Java のインストール・ページが開くので，画面に従って Java のインストールを行ってください．

Java のインストールが終わったら再度インストーラを起動して，STM32CubeMX のインストールを行います．

図 1-2　STM32CubeMX のインストール

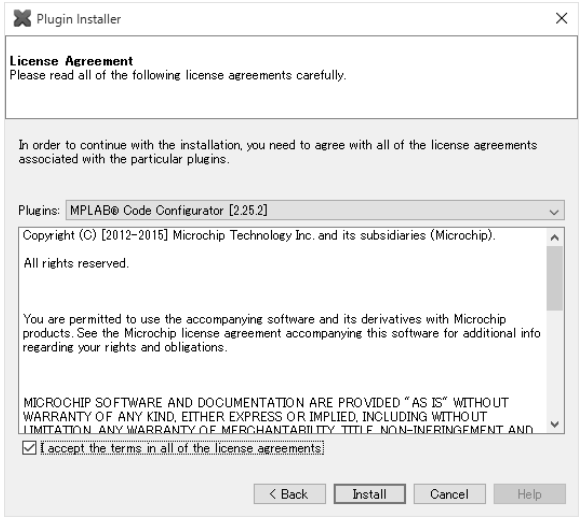


図 1-3　プラグインのインストーラ

PIC16F 開発環境のインストール

PIC16F の開発環境では MPLAB X IDE と XC8，それに Code Configurator をインストールします．どのツールもマイクロチップ社のウェブ・ページからダウンロードすることができます．付属 DVD-ROM にも収録されています．

最初に MPLAB X IDE をインストールします．MPLAB X IDE のインストールは次の手順で行います．

インストール・プログラムは，次のリンクからダウンロードすることができます．

http://www.microchip.com/pagehandler/ja-jp/family/mplabx/

Windows 用の MPLAB X IDE のインストーラをダウンロードして実行してください．MPLAB X IDE のインストール画面は，付属 DVD-ROM を参照してください．

MPLAB X IDE のインストールは統合開発環境のインストールのみで，コンパイラはインストールされません．別途コンパイラをインストールする必要があります．ここでは，マイクロチップ社の XC8 というコンパイラをインストールします．

MPLAB X IDE のインストールが終わると，ブラウザが起動しマイクロチップ社のウェブ・ページが表示されます．

このページの下，画面左側に，XC8 の Windows 版のダウンロード・リンクがあるので，これをクリックして XC8 をインストールします．

インストールは画面に従って［Next］ボタンで進めて行けば完了します．インストール画面は付属 DVD-ROM を参考にしてください．

なお，マイクロチップ社のコンパイラにはマイコンの種類によって，XC8，XC16，XC32 というコンパイラが用意されており，それぞれ 8bit，16bit，32bit のマイコン用となっています．PIC16F シリーズは 8bit マイコンなので XC8 を使用します．

ライセンスに関しては，Free 版，Standard 版，PRO 版があります．ここでは，無償の Free 版を使用します．

Code Configurator のインストール

最後に Code Configurator をインストールします．Code Configurator はプラグインとなっていて，インストールは MPLAB X IDE を起動した状態で行います．

まず，先にインストールした MPLAB X IDE を起動します．MPLAB X IDE が起動したら次の手順で Code Configurator をインストールします．

1. 「Tools」メニューを選択し「Plugins」を選択．
2. プラグインのウィンドウが開くので「Available Plugins」タブを開く．
3. プラグインのリストの中から「MPLAB Code Configurator」を選択（**図 1-3**）．
4. 左下の「I accept...」のチェック・ボックスにチェックを入れる．
5. ［Install］ボタンを押すとプラグインのインストーラが起動．
6. 画面に従ってインストールする．

LPC 開発環境のインストール

LPCXpresso IDE は次のリンクからダウンロードできます．付属 DVD-ROM にも収録されています．

http://www.lpcware.com/lpcxpresso/download

インストール画面は付属 DVD-ROM を参照してください．途中，いくつもドライバのインストールの確認画面が出ますが，すべてインストールしてください．

LPCXpresso IDE をインストールし LPCXpresso IDE を起動すると，最初にワークスペース・フォルダの確認画面が表示されます．

ここで［OK］ボタンを押してそのまま進めると**図 1-4** のような画面が表示されます．この画面は，まだ LPCXpresso IDE がアクティベーションされていないという警告なので，画面に従い，「HELP」-「Acivate」-「Create Serial number and resister(Free Edition)...」を開きます．

シリアル番号が表示され［OK］ボタンを押すと，LPCXpresso のウェブ・サイトに接続されます．

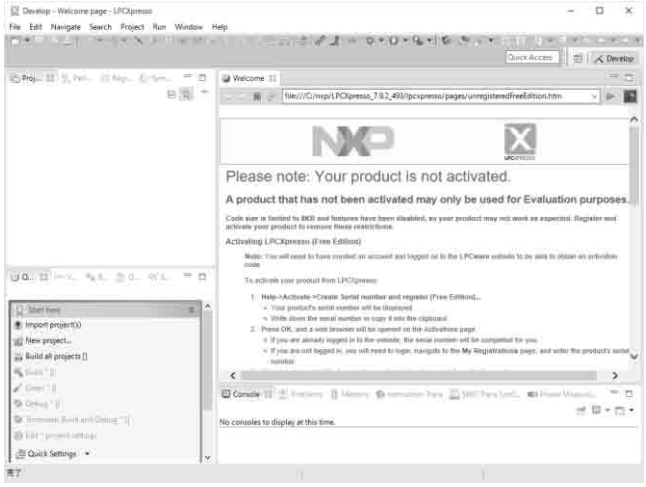


図 1-4　LPCXpresso IDE の起動画面

LPCXpresso IDE のアクティベーションを行うにはアカウントを作成してログインする必要があります．すでにログイン・アカウントを持っている場合は「Login」を選択し，まだ持っていない場合は「Register」を選択して新規にアカウントを作成してください．

アカウントを作成してログインすると，シリアル番号が自動で入力されるので，［Register LPCXpresso］ボタンを押して登録を完了します．

登録が完了すると，確認画面が表示されインストールは完了です．

第2章　I/O 制御ひな型プログラムの作成

ワンチップ・マイコンは多くのモジュールを内蔵していますが，使用できるピンは限られているため，一つのピンに複数の機能を割り当てています．どの機能を使用するかはソフトウェアで選択できるようになっています．

例えば，STM32F103RB（以降，STM32F103 と省略）の場合，PB14 は SPI，USART，タイマと GPIO を兼用していて，SPI，USART，タイマを使用しない場合はこのピンを GPIO として使用し，より多くの GPIO ピンが利用できるようになっています．

このような機能は便利ですが，マイコンでプログラムを作成する場合に最初に引っかかる問題の一つでもあります．特に初心者の場合や初めてそのマイコンを利用する場合は，このピンの割り当てをまちがえてしまい，作ったプログラムが動作しないということがよくあります．

PIC16F では一部の I/O ピンはデフォルトでアナログ入力になっていますが，多くの場合最初にテストするのは GPIO を使って LED を ON/OFF するようなプログラムです．たまたま LED を接続したピンがデフォルトで GPIO になっていないと，思うように LED が制御できずパニックになってしまうということがあります．

また，I/O ピンの機能の割り当て方法はマイコンごとに異なっているので，使い慣れていないマイコンの場合も同様な問題で頭を悩ますことになります．

そこで，ここではまず MPU トレーナ用 I/O 制御ひな型プログラム（フレームワーク）を作成し，MPU トレーナに合わせた I/O 設定までをフレームワークで行い，以降は使用する内蔵モジュールの制御に集中できるようにします．

Nucleo 用フレームワークの作成

STM32CubeMX でピン機能，内蔵モジュールを設定

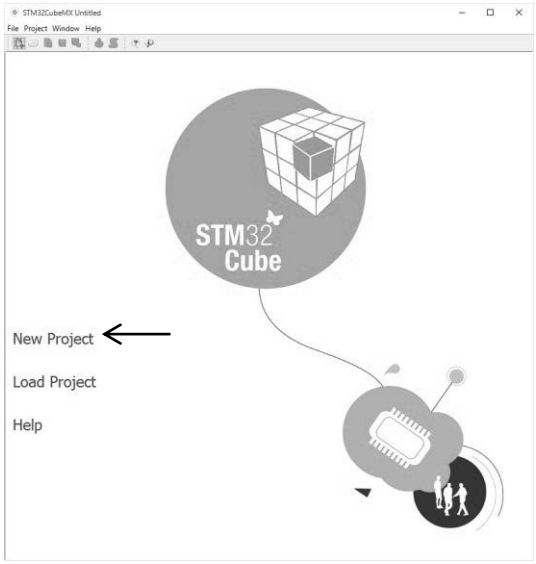
前述のように I/O ピンの設定は面倒でまちがえやすい作業なのですが，ST 社では STM32CubeMX という便利なツールを公開しています．これを利用すると I/O ピンの設定を GUI を使って簡単に行うことができます．

STM32CubeMX では STM32 の各種マイコンのほか，Nucleo などのボードを選択してピン・アサインを行うことができます．

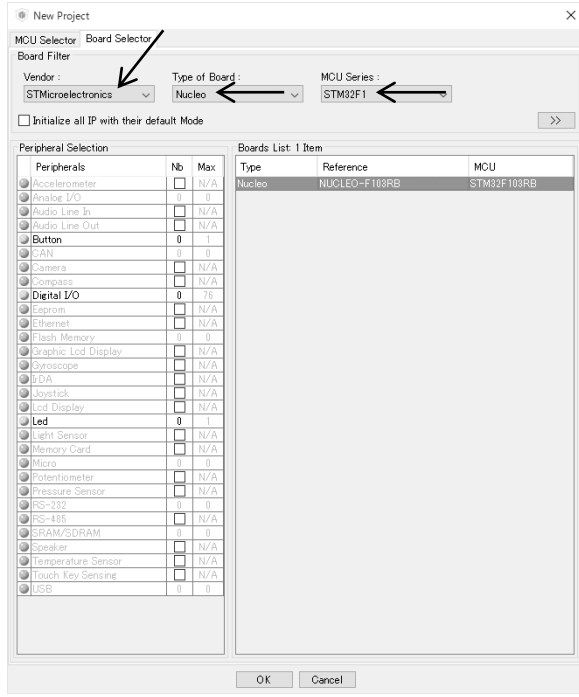
ボードを選択した場合は，搭載されているマイコンに選択したボードの内蔵モジュールが設定されるので便利です．

ターゲット・ボードの選択

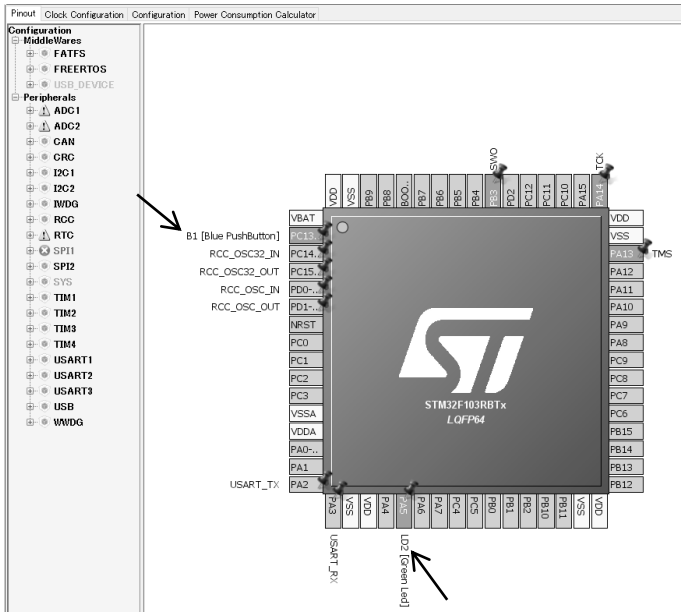
図 2-1（a）は STM32CubeMX の起動画面です．STM32CubeMX が起動したら，「New Project」を選択すると**図 2-1（b）**のような「New Project」ダイアログが表示されます．



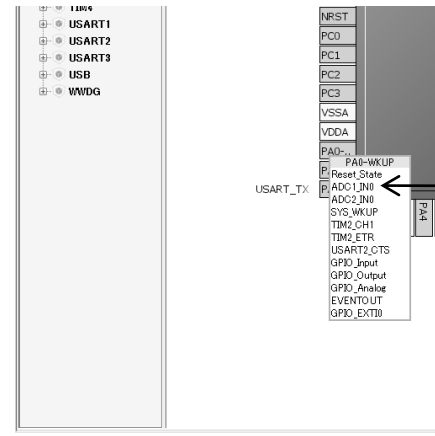
(a) STM32CubeMX の起動画面



(b) New Project ダイアログ



(c) ピン・アウトの設定



(d) PA0 の I/O 機能の設定

図 2-1　STM32CubeMX の使い方

「Board Selector」タブを開き，図のように「Vendor」に「STMicroelectronics」，「Type of Board」に「Nucleo」，「MCU Series」に「STM32F1」を選択し，「Boards List」から「Nucleo STM32F103RB」を選択して［OK］ボタンを押します．ほかのシリーズを選択する場合も同様の手順で行います．

ボードを選択すると，**図（c）**のようにピン・アウトの設定画面が表示されます．この画面では，あらかじめいくつかの I/O ピンが設定されていますが，これが Nucleo STM32F103RB の設定となっています．例えば，このボードには 1 個のユーザ用スイッチと 1 個のユーザ用 LED が接続されていますが，これはそれぞれ PC3 と PA5 に接続されていることが分かります．

I/O ピンの機能設定

I/O ピンの機能設定の方法について，PA0 を ADC1 の CH0 入力にする場合を例に説明します．

まず，**図（d）**のように画面上の PA0 のピンを左クリックします．ピンを左クリックすると図のようにポップアップ・メニューが表示され，設定可能な機能の一覧が表示されます．ADC1_IN0 を選択すると PA0 が ADC1_IN0 に設定されます．

このままでもよいのですが，さらに便利な機能としてユーザ・ラベル機能があります．

PA0 は，MPU トレーナでは AD1 という信号に接続されます．そこで，このピンに“AD1”というラベルを付けておくと，あとからこの画面を見たときにすぐに接続先が分って便利です．

ユーザ・ラベルを付けるには PA0 のピンをマウスで右クリックし，ポップアップ・メニューから「Enter User Label」を選択します．

図（e）のような画面が表示されるので，ラベル名を“AD1”として Enter を押すと，**図（f）**のように PA0 の信号名が“AD1”となります．

同様にして，MPU トレーナのピン・アサインをすべて設定した状態を**図（g）**に示します．

内蔵モジュールの機能設定

図（g）ではいくつかのピンがオレンジ色で表示されていますが，これは内蔵モジュールの機能設定が完了していないことを表しています．

この状態では I/O ピンは指定した内蔵モジュールに接続されていますが，その内蔵モジュールの機能が設定されていないため，正しく動作しない状態となっています．

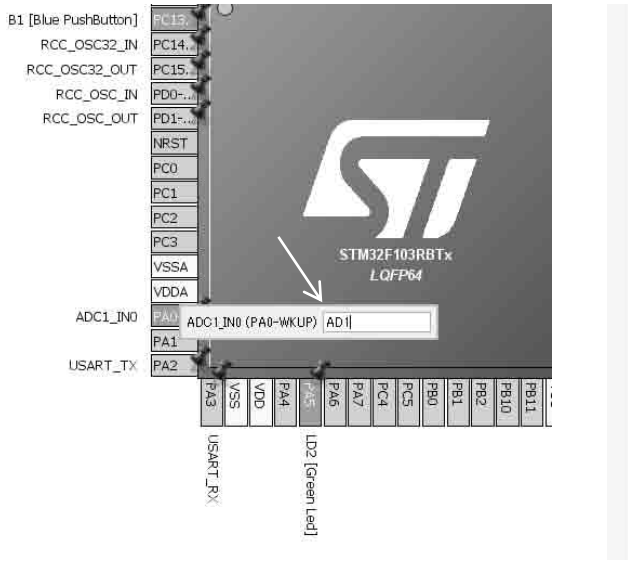
例えば，画面上部中央の PB4 のピンは PWM の出力端子 PWMOUT として設定されています．PWM の出力はタイマの機能の一つですが，このタイマの設定をしていないため警告の意味でオレンジ色の表示となっています．

PWMOUT は，TIM3 の CH1 の機能の一つを利用しているので，画面左側の内蔵モジュールのツリーから「TIM3」を開き，「Channel1」を**図（h）**のように「PWM Generation」に設定します．

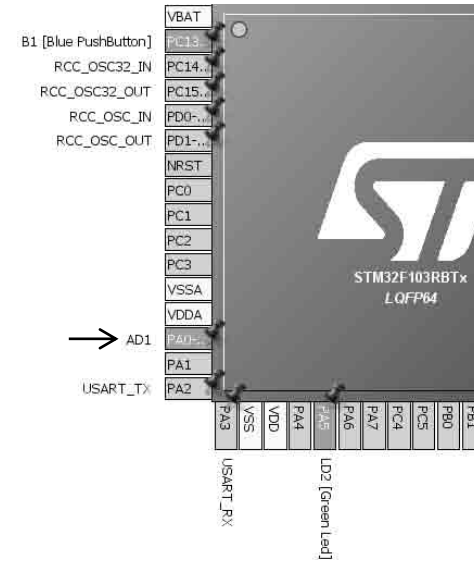
同様に「TIM1」の「Channel2」を「Input Clock」に，「SPI2」を「Full Duplex」に設定します．

通信パラメータと割り込みの設定

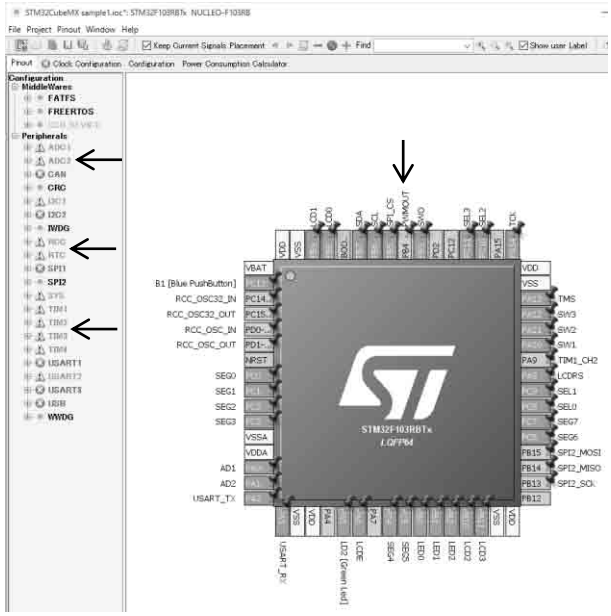
通信パラメータの初期設定は「Configuration」タブで行います．「Configuration」タブを開いて「Connectivity」から「USART2」をクリックすると，**図（i）**のようなダイアログが開きます．



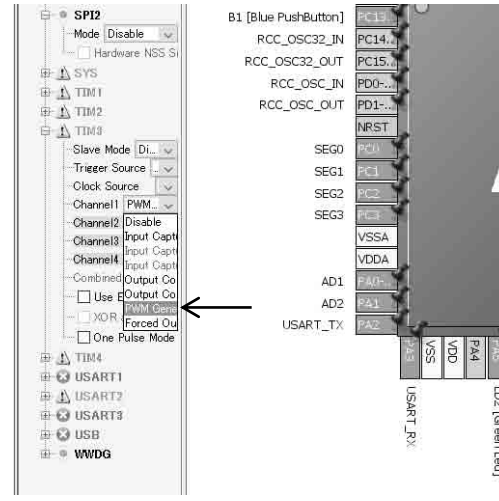
(e) ユーザ・ラベルの設定



(f) PA0 のラベルを設定完了

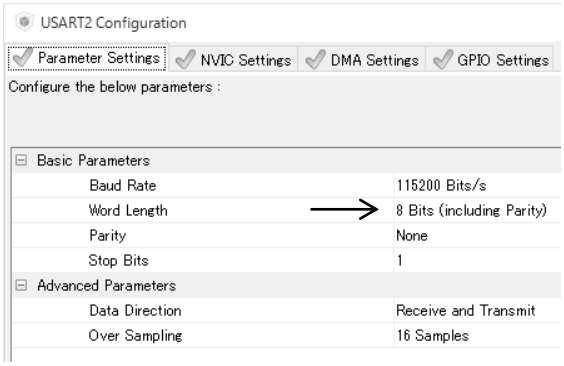


(g) MPU トレーナのピン・アサイン設定後の画面

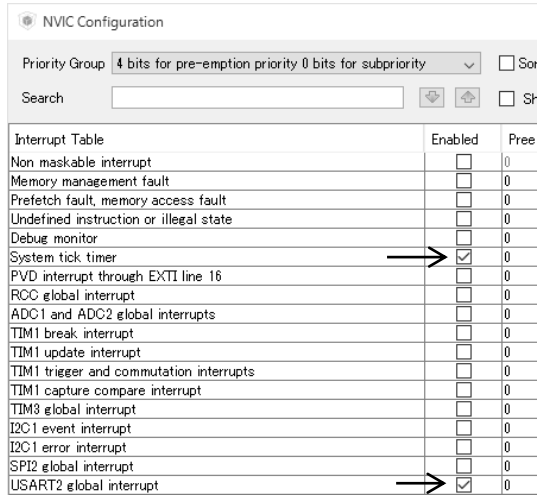


(h) TM3 Channel1 を PWM Generation に設定

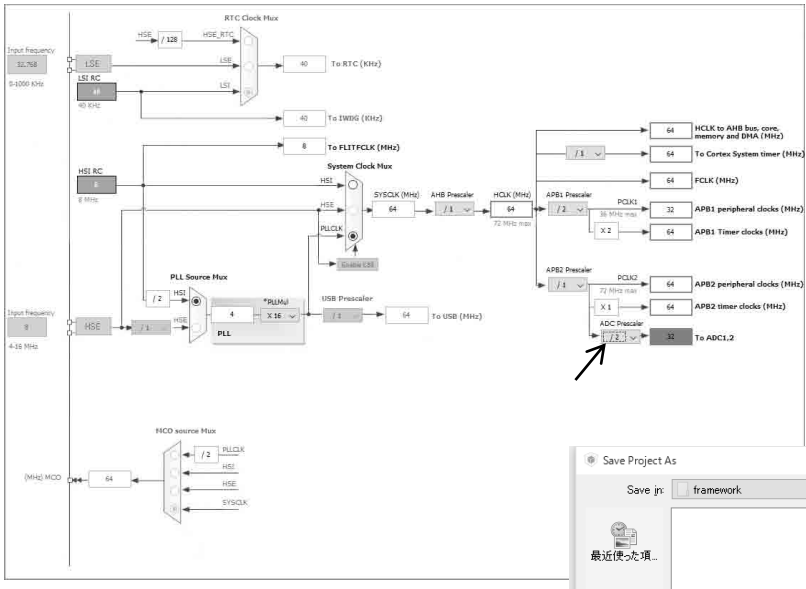
図 2-1　STM32CubeMX の使い方（つづき）



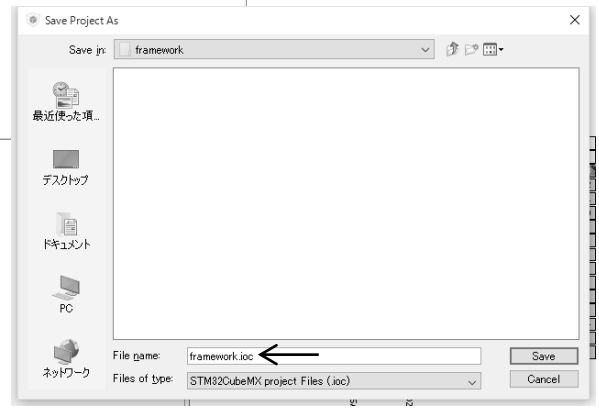
（i） USART2 の設定ダイアログ



（j） 割り込みの設定ダイアログ



（k） 「Clock Configuration」タブの画面



（l） STM32CubeMX プロジェクトの保存

図 2-1　STM32CubeMX の使い方（つづき）

ここで，ボー・レートなどの必要なパラメータを設定します．デフォルトでは「Word Length」が「7bit」になっているので「8bit」に設定します．

割り込みを使用する場合は，同じように「Configuration」タブから「System」の［NVIC］ボタンを押して設定します．**図**（j）の画面で割り込みを使用するデバイスにチェックを入れます．

ここでは「USART2」の割り込みを有効にします．また，「System tick timer」もデフォルトで有効になっているのでそのまま有効にしておきます．

クロックの設定

ADC を有効にすると「Clock Configuration」タブに赤いマークが表示されます．「Clock Configuration」タブを開くと，**図**（k）のように ADC のクロックが赤い色で表示されています．

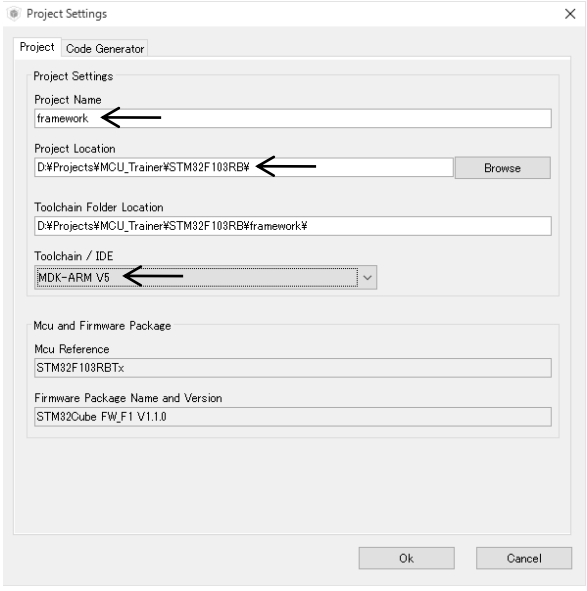
これは ADC のクロックが上限を超えているためなので，「ADC Prescaler」の設定を「1/8」に設定して ADC のクロックを 8MHz に変更します．

MDK-ARM プロジェクトの生成

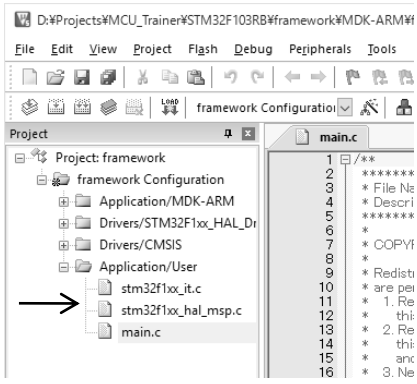
設定が完了したら，STM32CubeMX のプロジェクトを保存しておきます．「File」メニューから「Save Project」を選択して，**図**（l）のように，適当なフォルダに“framework”という名前でプロジェクトを保存します．

ここで保存したプロジェクトは，STM32CubeMX のプロジェクトです．このファイルは GPIO の設定や各種パラメータを後から変更したい場合に使用します．

これで設定はひととおり完了したので，最後に MDK-ARM のプロジェクト・ファイルを生成します．「Project」メニューから「Generate Code」を選択すると，**図**（m）のようにプロジェクトの保存先の



(m) プロジェクトの保存先のダイアログ



(n) ロードされたプロジェクト

図 2-1　STM32CubeMX の使い方（つづき）

ダイアログが表示されます．ここでは，MDK-ARM のプロジェクトの保存先，プロジェクト名を指定します．

STM32CubeMX では MDK-ARM 以外のコンパイラもサポートしているので，使用するコンパイラも指定する必要があります．コンパイラの指定は「Toolchain /IDE」のドロップダウン・リストで行い，図のようにツールチェインとして「MDK-ARM V5」を選択します．設定が完了したら［OK］ボタンを押すと MDK-ARM のプロジェクトが自動で生成されます．

HAL ライブラリのインストール

STM32CubeMX で生成するプロジェクトは，HAL というライブラリを使用します．HAL のライブラリはマイコンのシリーズごとに用意されています．このライブラリは必要に応じて自動でインストールされます．初めて STM32F103 のプロジェクトを生成する際は，この STM32F1 シリーズ用の HAL ライブラリのダウンロードが始まるため少々時間がかかりますが，次回からはダウンロードは行われません．

ライブラリはローカルからもインストールできます．ライブラリは付属 DVD-ROM に収録されています．「Help」 - 「Install New Libraries」を選び，「New Libraries Manager」の［From Local］ボタンを押して表示されるダイアログで，サンプルの場合，stm32cube_fw_f1_v110.zip を指定します．

自動生成されたファイル

MDK-ARM のプロジェクトが生成されると，プロジェクトを開くかどうかを聞いてきます．ここでプロジェクトを開くと MDK-ARM が起動し，**図**（n）のように生成したプロジェクトがロードされます．

生成されたプロジェクトは図のように四つのフォルダに分類されています．ユーザ・プログラムは「Application/User」フォルダにあり，それ以外のフォルダはライブラリ・ファイルとなっています．

ユーザ・プログラムには，次の三つのソース・ファイルがあります（付属 DVD-ROM に収録）．

main.c

main.c は main()を含むソースです. 基本的にはユーザ･プログラムは main.c のみを修正します. main()は設定した内蔵モジュールを初期化するコードが含まれているので，初期化後の whlie 文を必要に応じて修正して，ユーザ独自の機能を実現します．

stm32f1xx_it.c

stm32f1xx_it.c は割り込み関連の設定が記述されています．使用する割り込みをユーザが使用できるようにするために，このファイルが自動生成されます．

stm32f1xx_hal_msp.c

stm32f1xx_hal_msp.c は内蔵モジュールの初期化コードから呼び出される関数が記述されています．この関数では，I/O のピン設定の変更やクロック設定，割り込みの設定を行っています．

以上で，Nucleo STM32F103RB のフレームワークは完成です．他の Nucleo ボードのフレームワークも同様の手順で生成することができます．

PIC16F 用フレームワークの作成

MPLAB X IDE には MPLAB Code Configurator というツールがあります．このツールは STM32CubeMX と同様に，GUI を使って内蔵モジュールの初期化プログラムを生成できるツールです．

STM32CubeMX はスタンドアロンのプログラムでしたが，MPLAB Code Configurator は，MPLAB X IDE のプラグインとなっており，プロジェクト・ウィザードの補助的なツールのようになっています．

ここでは，ターゲット・デバイス PIC16F1789 のフレームワークを作成します．

プロジェクトの作成方法

最初に，プロジェクト・ウィザードを使い，PIC16F1789 のプロジェクトを作成します．

プロジェクト・ウィザードの操作手順を図 2-2（a）~図 2-2（g）に示します．

1. MPLAB X IDE を起動したら「File」メニューから「New Project」を選択してプロジェクト・ウィザードを起動する［図（a）］．カテゴリから「Microchip Embedded」を選択し，「Projects」からは「Standalone Project」を選択する．
2. ［Next］ボタンを押すとデバイスの選択画面になるので，ターゲット・デバイスとして「PIC16F1789」を選択する［図（b）］．
3. サポート・デバッグ・ヘッダの選択［図（c）］は「None」のままで，「Select Tool」では「PICkit3」を選択する［図（d）］．
4. 「Select Compiler」では「XC8」を選択する［図（e）］．
5. 最後にプロジェクトの名前と保存先を入力する［図(f)］．プロジェクト名は“framework”とする．「Encoding」は「Shift_JIS」に変えておくと日本語のコメントが入れられる．

Code Configurator でピン機能，内蔵モジュールを設定

プロジェクト・ウィザードでプロジェクトが作成されると，図（g）のように framework プロジェクトがオープンされた状態となります．この時点ではまだ空のプロジェクトで，実際のソース・ファイルはまだ何も作成されていません．そこで，次に MPLAB Code Configurator を使って MPU トレーナ用のフレームワークを生成します．

MPLAB Code Configurator の起動は図 2-3 のように，「Tools」メニューから「Embedded」-「MPLAB Code Configurator」を選択します．

Code Configurator を起動すると，図 2-4 のような画面となります．Code Configurator では内蔵モジュールごとに設定を行います．Code Configurator 画面の左上には「Project Resources」というペインがあり，ここに登録した内蔵モジュールが表示されます．図 2-4 では初期状態として，「System」というリソースのみが登録されている状態となります．

「Project Resources」の下には，デバイス・リソースのペインがあります．このペインから追加するリソースを選択してプロジェクト・リソースに追加します．

画面中央は選択されたリソースの設定ペインで，ここでリソースの設定を行います．図 2-4 ではシステム・リソースの設定画面なので，クロックの設定やコンフィグレーション・ビットの設定を行います．

定番! ARMキット&PIC用

Cプログラムでいきなりマイコン制御 [DVD-ROM付き]

Rapid operations & prototyping with programmed C source code for ready-made ARM/PIC CPU boards

本書に関するQ&A

1 本書で何が学べますか?

ワンチップ・マイコンの内蔵モジュールの基本的な使い方が学べます.

2 内蔵モジュールとは何ですか?

主にワンチップ・マイコンの入出力回路のことです. GPIO, タイマ, A-Dコンバータ, UARTなどを指します.

3 周辺デバイスや周辺モジュールとは違いますか?

同じです. CPUにPPIなどの周辺LSIを繋いでマイコン・システムを設計していた名残で周辺デバイスという言い方もされています. 本書では1チップに内蔵されているという意味から内蔵モジュールと呼んでいます.

4 ソフトウェアとハードウェア, どちらも学べますか?

マイコンを利用するにはプログラムを作成できるようになることが重要と考え, ソフトウェア技術を優先しています.

5 Nucleoとは何ですか?

STマイクロエレクトロニクスのARM CortexマイコンSTM32搭載ボードです.

6 LPCXpressoとは何ですか?

NXPセミコンダクターズのARM CortexマイコンLPC搭載ボードです.

7 PIC16F1789I/P, PIC16F1939I/Pとは何ですか?

マイクロチップ・テクノロジーのPIC16Fシリーズの40ピンDIPマイコンです.

8 MPUトレーナとは何ですか?

Nucleo, LPCXpresso, PIC16F1789I/P, PIC16F1939I/Pのいずれかを装着して使用するマイコン制御学習用ベース・ボードです. 本書のサンプル・プログラムはすべてMPUトレーナで動作確認しました.

9 動作確認済みのNucleo, LPCXpressoはどれですか?

NUCLEO-F103RB, NUCLEO-F072RB, NUCLEO-F401RE, LPCXpresso 1347, LPCXpresso 1769, LPCXpresso 11C24です.

10 Nucleo, LPCXpresso, MPUトレーナはどこで買えますか?

本文の入手先例をご覧ください.

11 本書で使用している開発ツールの動作環境は?

Windows 7/8.1/10を推奨します.

12 Arduino, mbedの環境は使えませんか?

本書では使用していません. 組み込み製品開発, 特に小規模では本書のC言語で一からクロス開発するケースが多いので, それになるべく近い環境を選択しています.