8ピンDIPでどこにでもぶち込める！
実験／研究／工作にピッタリ！

ハードウェア・セレクション

挿すだけ！ ARM 32ビット・マイコン のはじめ方

中村 文隆 著

CQ出版社

第1章 誕生DIP8ピンARMマイコン

LPC810

● DIPのARMとは？

LPC810は，NXPセミコンダクターズ社のARMマイコン・シリーズの中で，もっとも小さな8ピン・パッケージの，DIPマイコンです(**図1**)．CPUコアは，12MHz（最大30MHz)動作のCortex-M0+で，4KBのフラッシュ，1KBのRAMを使用することができ，ROMに置かれたUART/I²C APIによる手軽なI/Oを利用することもできるようになっています．

動作電流は，30MHz動作時で3.3mA，内蔵のRCオシレータ(12MHz)を使った場合は，1.4mAと小さく，電源電圧も3.3Vのため，電池での駆動にも向いています．パッケージの小ささや，1個100円程度の安価な販売価格とあわせて，ちょっとした電子工作を楽しむには，まさにうってつけのマイコンです．

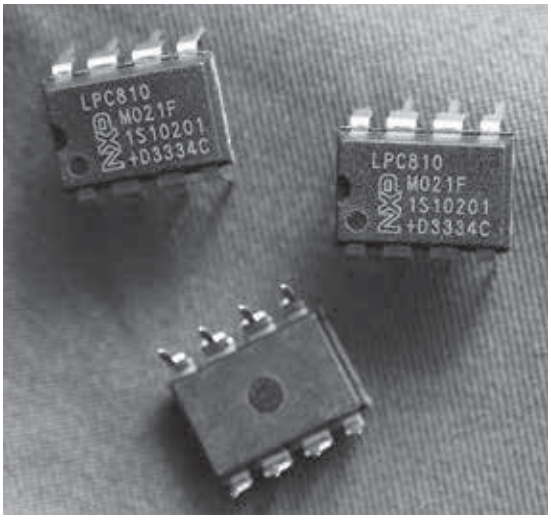


図1　LPC810のパッケージ

現状，NXP社のCortex-M0+のラインナップで，DIPパッケージのものは，8ピンのLPC810のみです．ピン数の多いDIPパッケージは，Cortex-M0コアDIP28ピン・パッケージのLPC1114となります．それ以外の，TSSOPや，SOパッケージのものは，IC単体を購入して個人で楽しむには，実装が煩雑なため，ボードとして提供されているものを使うことが多くなりそうです．

LPC810は，8ピンのパッケージで，電源とGNDのピンの二つを除くと，I/Oとして使えるピンは，6本と少ないものの，I/O機能のピン・アサインは，スイッチ・マトリクスという仕組みでプログラムから動的に変更することができます．やりたいことに対して，どうしてもピン数が不足するようなケースでは，LPC1114やボードを使う[1]ことになると思われます．しかし，手軽なプロトタイピングや，小規模な開発などにおいては，LPC810は最初に検討してみる価値のある選択肢だといえるでしょう．

また，LPC810には，有限状態機械[2](finite state machine)をハードウェアで実装したSCT（State Configurable Timer)も組み込まれています．このSCTは，NXPのマイコン・ラインナップの中では，LPC8xx，LPC18xx，LPC43xxの三つの系列でのみ使えるもので，そのエントリ・モデルとしてのLPC810は，用途によっては魅力的な選択肢になり得ます．

1　外付けのI/O拡張回路を使う方法もある．しかし，ほとんどの場合，I/O拡張回路のほうがLPC810よりも物理的なサイズが大きくなってしまう上，LPC1114を単体で購入するよりも高額になり，配線の手間もかかる．また，LPC810には，5ビットのアナログ・コンパレータが2系統あるものの，ソフトウェアでの制御が必要であり，A-Dコンバータが必要なケースでは，10bit×6chのハードウェアA-Dコンバータを持つ，LPC11140を使うほうが素直である．
LPC8xxシリーズには，後述のSCTや高速GPIOがある点が，LPC11xxシリーズに対してのアドバンテージであり，それらが必須の要求条件で，かつDIPパッケージを使った単体での製作を行いたい場合には，LPC810が有力な検討対象になり得る．

2　有限オートマトン(finite automaton)とも言われる．

さらに，LPC810 の CPU コアである Cortex-M0+ の特徴の一つに，GPIO が CPU 直結されている点があげられます．バス経由ではないため，GPIO へのアクセスを高速に行うことができます．また，ディジタル・ピンのセット / リセットも効率的に行うことができるようになっていて，比較的単純ではあるけれど，高速な I/O が必要な場合には，LPC810 が良い選択肢になることもあるでしょう．

本書では，まず LPC810 のハードウェアとソフトウェアについて概観したあと，USB-シリアル・インターフェースを使った手軽な開発環境を説明し，LPC810 単体での機能説明と活用例を中心に解説し，LPC810 の楽しさをみていきたいと思います．開発環境の USB-シリアル・インターフェースを使うと，LPC810 をパソコン I/O インタフェースとしても機能させることができるので[3]，それについても事例を紹介してみます．また，スマート・フォンとの音響通信についても取り上げます．

なお，以下では，LPC810 に関する，次の二つのマニュアルを適宜参照しながら説明を行います．

LPC800 User manual, Rev.1.3-22 July 2013
http://www.nxp.com/documents/user_manual/UM10601.pdf
LPC81xM 32-bit ARM Cortex-M0+ microcontroller, Rev.4.2-10 December 2013
http://www.nxp.com/documents/data_sheet/LPC81XM.pdf

以降の解説では，上記のそれぞれを，UM，XM と略記します．例えば，「UM p.264」は，UM10601.pdf の 264 ページ，「XM pp.9-11」は，LPC81XM.pdf の 9 ページから 11 ページと読み替えてください．

● スペック

ハードウェア

現在，店頭で販売されている，LPC810 の正式な型番は，LPC810M021FN8 で，8 ピンの DIP パッケージに収められています．

おもにプログラムのバイナリ・コードを保存する Flash の領域が 4kB，実行時のワーキング・メモリに使用される，SRAM が 1kB，それぞれ搭載[4]されています．

I/O は，おもなところでは，シリアル通信の USART[5] が 2 系統，I²C が 1 系統，SPI が 1 系統，アナログ・コンパレータが，2IN/1OUT 系統，GPIO が 6 系統，SCT が 4 系統，それぞれ利用可能です．

LPC810 のパッケージは，DIP8 で，電源と GND に使用する二つのピンを除くと，I/O に使用できるピン数は最大で 6 ピンしかないため，必要に応じてこれらの I/O 機能のピン・アサインを切り替えて使用できるようになっています．

I/O 関係の概略図は，**図 2** のようになっています．バスは，CPU と主記憶(SRAM/ROM)，さらに FLASH も AHB(Advanced High-Performance Bus) と呼ばれる高速バスに接続されています．AHB-LITE は，本来の AHB からバス上のデバイス間のアービトレーション(調停)機能を省略したもので，CPU がマスタとなり，他の AHB-LITE 上のデバイスが slave となることで，調停を省略することができるため，設計が簡略化されているものです．

周辺 I/O は，ブリッジを介して APB（Advanced Peripheral Bus)上に接続されています．APB では，割り込みに IRQ（Interrupt ReQeust)を使い，NVIC (Nested Vectored Interrupt Controller)で管理されています．

I/O のうち，GPIO は，CPU に直接接続されているため，高速な I/O が期待できます．また，SCT（State Configurable Timer)[6] は，CPU とは独立に稼働するブロックになっているため，こちらもパフォーマンス的には期待できる構成となっています．それ以外の主要な I/O は APB に接続されています．

図 2 にあるように，I/O 関係の信号線は多数あるものの，DIP8 ピン・パッケージでは電源と GND を除いた最大 6 ピンまでしか物理的な I/O には使用できないため，スイッチ・マトリクスと呼ばれるブロックで，

---

3 この用途の場合は，上位 CPU をボード化した製品を使うほうが，金額，物理サイズ，ソフトウェア開発などを総合したトータル・コストを考慮すると，良い選択になる．LPC810 でも実現できるという事例の提示として考えていただきたい．

4 OS が動いているわけではないので，SRAM が不足で，Flash との間でページングとしたければ，自分でその処理コードを書く必要がある．

5 USART は Universal Synchronous and Asynchronous Receiver and Transmitter の頭文字を並べたもので，同期式(Synchronous)と非同期式(Asynchronous)のどちらかの方式でシリアル・データ転送を行う機能．すべてのシリアル通信を USART と呼ぶわけではなく(たとえば USB や I2C もシリアル通信であるが USART には含まれない)，USART に属する電気的な規格や通信プロトコルがいくつか定められている．
ホビー用途では，多くのパソコンで利用可能な，非同期式の RS-232C のインタフェースとして使用する機会が多く，その場合は，Synchronous を省略して UART と表記されることもしばしばある．パソコンとの RS-232C での通信のみを考えている文脈では，RS-232C = UART = USART として，しばしば混在した用法がみられるが，概念の包含関係としては RS-232C ⊂ UART ⊂ USART である．本書では，LPC810 の USART は RS-232C としてのみ使用するので，RS-232C，もしくは UART と表記する．

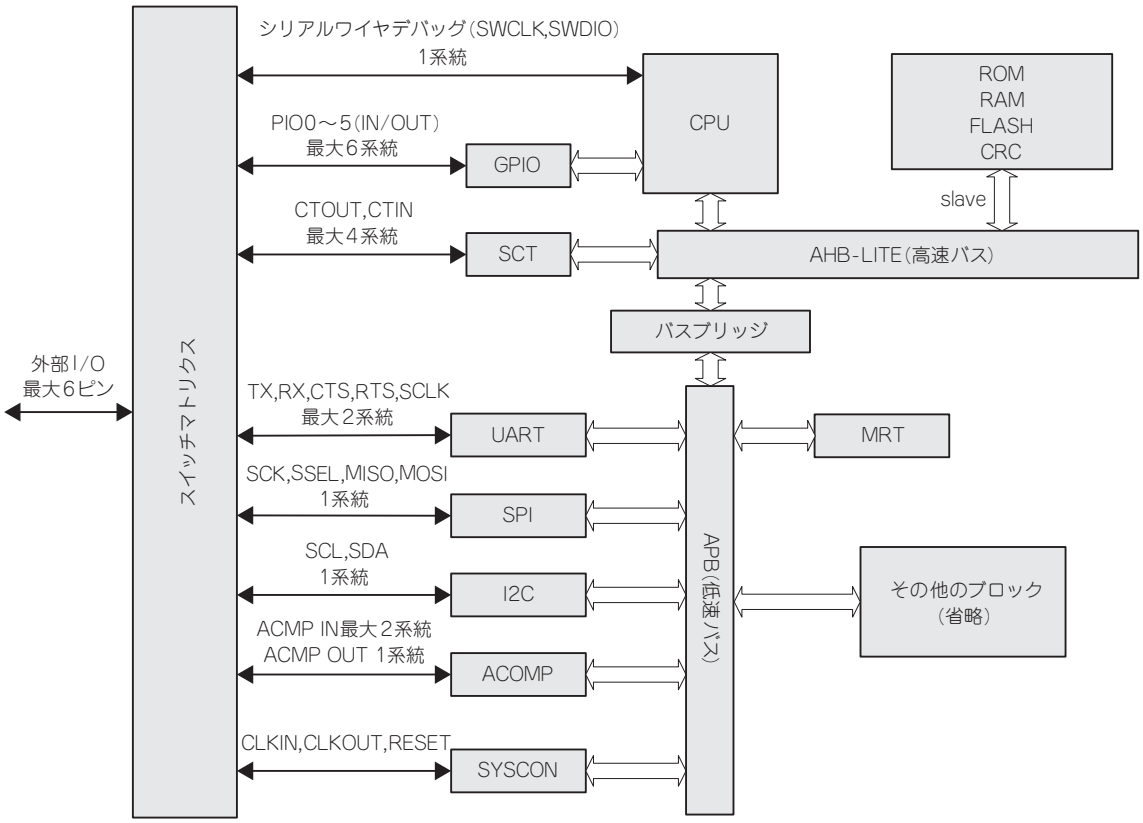


図2　LPC810 入出力関係概略図

信号を選択して I/O に用いるようになっています．

CPU は，Corrtex-M0+コアで，動作周波数は最大 30MHz ですが，DIP パッケージに内蔵されている RC オシレータ(Internal RC = IRC オシレータ)は 12MHz です．初期状態では IRC オシレータからのクロックが供給され，12MHz で動作します．電源電圧は 3.3V で，アクティブ電流は，110μA/MHz のため，12MHz 動作時の消費電流は，1.4mA ほどで，スペックからすればかなりの低消費電力で動作しています．クロック周波数は内部の PLL を使用してクロック・アップすることも可能です．

なお，LPC800 シリーズの他のチップでは，外部接続のクリスタルを使用することも可能ですが，LPC810 では外部のクリスタルを接続するピンをアサインすることができないため，内蔵クロックか，CLKIN からの直接のクロック供給のどちらかでの動作となります．CLKIN は，**図 2** からわかるように APB 経由で，活用できる場面は限定的[7]と考えるべきでしょう．現実的には，LPC810 の場合，IRC オシレータからのクロックをそのまま用いるか，必要であれば内蔵の PLL でクロック・アップするという使い方になります．

ソフトウェア

LPC810 のソフトウェアは，LCXPresso という統合開発環境(IDE = Integrated Development Envrionment)を用いて，C 言語で記述するスタイルで作成することが一般的です[8]．統合環境で作成したプログラムのバイナリ・コード(ユーザ・コード)は，UART（シリアル通信）経由での ISP(In-System Programming)で LPC810 に書き込みます．通常，LPC810 には，OS は存在せず，起動後は，Boot ROM のコードがまず実行されます．Boot ROM のコードの基本的な動作は，**図**

6　上位機種では，最大 32 の state を定義できるが，LPC810 では定義できる状態数は，一つのカウンタにつき，二つまでである(16 ビット ×2 の使い方であれば各系統に二つずつ)．SCT の編集ツール上は，U_ALWAYS（もしくは H_ALWAYS と L_ALWAYS)という擬状態(pseudo state)が定義でき，それを加えると見かけ上，三つに見えるが，これはステート・マシンがどの state にいるかに関係なく発生するイベントを表すためのシンボルで，通常の状態として使うことはできない．

7　CLKIN からのクロックを PLL でクロック・アップすることも可能で，LPC810 内での分周では，どうしてもまかないきれないクロックが必要な場合などが考えられる．

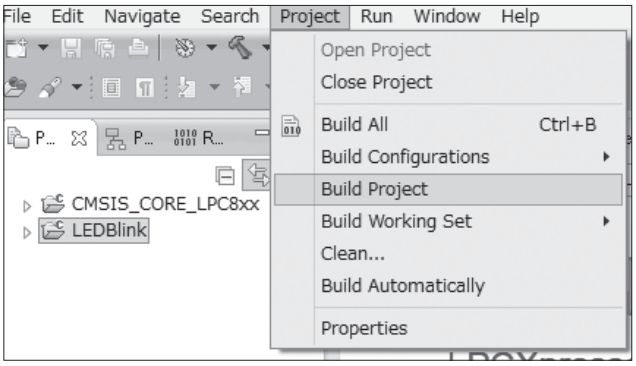


図 51　プロジェクトをビルドする

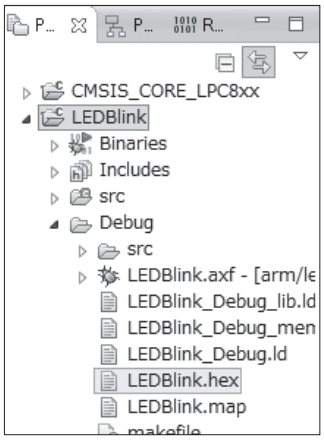


図 52　Debug フォルダの中に .hex ファイルができている

ビルドと HEX ファイル生成

ここまでくれば，あとはプロジェクトをビルドし，FlashMagic で LPC810 に書き込むための HEX 形式のファイルを生成するだけです．

プロジェクトのビルドは，**図 51** のように，ビルド対象のプロジェクトを選択した状態で，メニューから Project → Build Project と選択します．ビルドは，ソース・ファイルをコンパイルし，ライブラリなどをリンクしてバイナリ・ファイルを作成する作業で，今の場合は，**図 43** で追加した，post build のコマンドがビルド後に実行され，バイナリ・ファイルから LPC810 転送用のインテル HEX 形式までが生成されます．

通常，プロジェクトには，Debug と Release の二つの設定がありますが，ウィザードで生成したプロジェクトの場合，デフォルトでは Debug 設定でのビルドが行われます．

このため，ビルドに成功すると，**図 52** のようにプロジェクト内の Debug[21] というフォルダの中に LEDBlink.hex というファイルが生成されています．この HEX 形式のファイルを Flash Magic を使って LPC810 に転送します．

● LED Blink プログラムの書き込みと実行

ISP モードの結線

ここまでで，LED 点滅プログラムのバイナリが，LPC810 に書き込むための HEX 形式で用意されたことになります．LPC810 にプログラムを書き込むには，パソコンからの USB-シリアルと LPC810 を**図 53** のように接続します．**図 53** のように，LPC810 のピン 5 を GND に接続した状態で再起動する（電源を入れ直すか，リセット[22] をかける）と，LPC810 が ISP（In-System Programming）のモードで起動し，そのときのピン・アサインは，図 53 のように，

ピン 2　　送信（U0_TXD）
ピン 8　　受信（U0_RXD）

となります．LPC810 側の送信（受信）をパソコン USB からの受信（送信）に，クロスの状態で接続することで，パソコン側からのプログラム書き込みができるようになっています．本書では，この方法で ISP モードに入れた LPC810 に，パソコンから USB-RS232C 変換を介してプログラムを書き込む方法で開発を進めていきます．

なお，プログラム書き込み時の ISP モードのピン・アサインは，Switch Matrix Tool でのピン・アサイン変更とは独立した話で，ピン 5 が GND に接続された状態で起動シーケンスを経過した場合は，書き込まれているプログラムが実行されずに ISP モードに入るため，LPC810 に書き込まれているプログラムでの

21 Release 設定に切り替えるには，LPCXpresso 内でプロジェクトを右クリックし，Build Configurations → Set Active と選択して，Release を選択すればよい．この切り替えは，インポートしたライブラリを含めてワークスペース内の関係するプロジェクトを，すべて Release に切り替える必要がある．また，ビルド時の HEX ファイル生成設定も，Release 側で改めて行う必要がある．
コード・サイズは，Release のほうが若干小さくなるが，LEDBlink の場合で HEX ファイルのサイズ差は 21 バイトで，そこまでシビアではない．

22 リセットは，ピン 1 のアサインが RESET（初期状態のピン・アサイン）である場合は，ピン 1 を GND に落としてから 3.3V にプルアップすることで行う．ピン 1 の機能は変更できるため，通常の起動後にプログラム内からピン 1 のアサインを RESET 以外のものに変更しているときには，この方法は使えない．
LPC810 は，ピン数に余裕がないため，RESET にピンを 1 本取られるよりは電源の再投入での再起動に割り切って使うのも一つの手であるといえる．

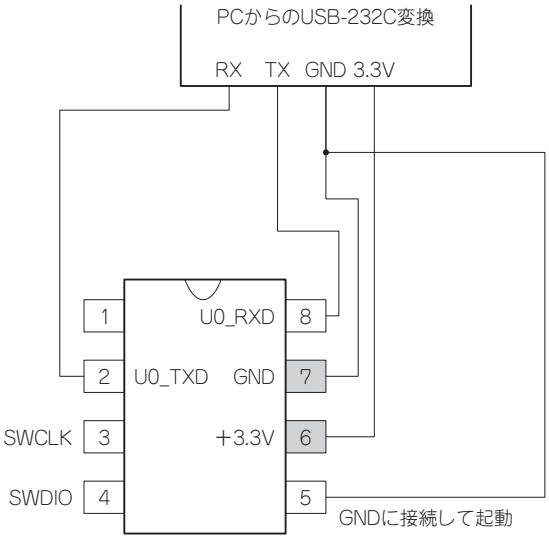


図 53　ISP でのプログラム書き込み時の接続とピン・アサイン(ISP モード起動時)

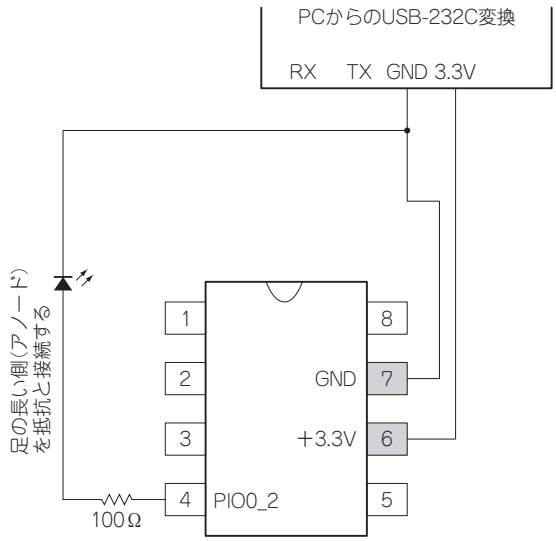


図 54　LED 点滅動作確認の接続とピン・アサイン(ユーザ・コード実行時)

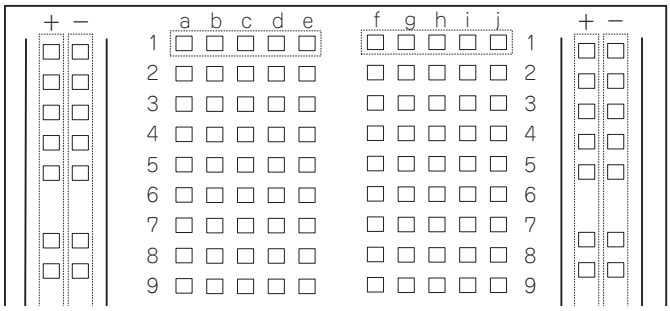


図 55
ブレッドボードの導通

ピン・アサイン変更とは無関係に，**図 44** ではなく，**図 53** のピン・アサインとなります．

LPC810 の電源電圧は 3.3V ですが，デフォルトの 12MHz 動作時でも動作電流が 1.4mA と小さいため，プロトタイピング時の電源は，パソコンからの USB − RS232C 変換アダプタから供給します．基本的な流れとしては，ブレッドボード上で検証用の回路を組み，プログラムを書き込んでテストしたうえで，必要であればユニバーサル基板やプリント基板上にシステムを組み上げる，という順序で作業を進めます．

LED 点滅動作時の結線

一方，プログラムの書き込みが終わって，LED の点滅動作をさせるときの接続は，**図 54** のようになります．今回作成した点滅プログラムでは，Switch Matrix Tool でピン 4 を PIO0_2 にアサインし，プログラムの中で DIR0 の PIO0_2 のビットを立てて，PIO0_2 を出力モードにしています．I/O で使用しているのは，このピン 4 だけなので，あとは，ピン 6 を 3.3V の電源に，ピン 7 を GND に接続するという三つのピンの接続となります．

PIO0_2 として使うピン 4 には，電流制限抵抗の 100Ω をつなぎ，抵抗の先に LED のアノード(A) = 長い方の足をつなぎ，LED のカソード(K) = 短い方の足を GND に接続します．

ブレッドボード上の実装

以上の**図 53** と**図 54** の接続を，ブレッドボード上に作成しておきます．よくみかける手ごろなブレッドボード上のホール(差し込み穴)の導通関係は，**図 55** のようになっています．

ブレッドボードには特に規格が定められているわけではないので，**図 55** とは異なる仕様のブレッドボードもあります．用意したブレッドボードの導通に確信が持てない場合は，テスタを使用するなどして，ホール同士の導通関係を確認しつつ作業を進めてください．

LPC810 の開発にあたっては，**図 53** の ISP モード

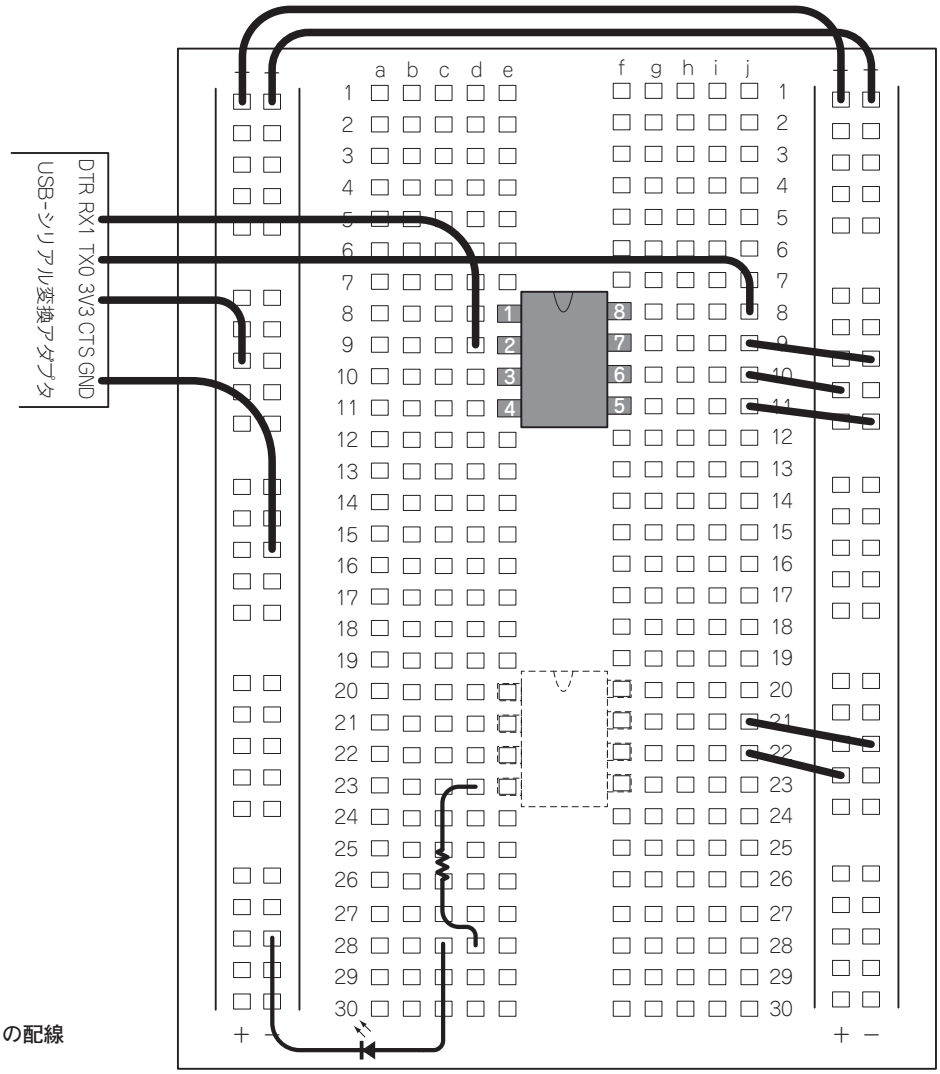


図 56
LED 点滅動作チェックの配線
（書き込み時の位置）

書き込み時と，**図 54** のユーザ・コード・モードでの実行時とを切り替えつつ，進めていく必要があります．

本書では，DIP8 ピンの IC ソケットに差し込んで，ゲタをはかせた LPC810 を，**図 56** と**図 57** のように，書き込み時と実行時で差し替えることで，この切り替えを行うことにします．

つまり，ブレッドボード上に，**図 53** と**図 54** の接続の両方を同時に用意しておき，差し替えを行うときは，USB-シリアル・アダプタから供給される電源，または GND のラインを一旦抜いて LPC810 の電源を落とし，差し替えが終わってから抜いたラインを戻すことで起動させるようにします．

図 56 と**図 57** は，LPC810 の差し込み位置が異なるだけで，そのほかの配線は同じものになっています．上部には**図 53** の状態の配線，下部には**図 54** の状態の配線を作ってあります．**図 56** と**図 57** の中で，USB-シリアル変換アダプタ，と表記されているものは，**図 9** のパーツです．なお，配線を組んでいくときには，USB-シリアル変換アダプタは，パソコンに接続しないようにするか，もしくは，USB-シリアル変換アダプタからブレッドボードへの 3.3V 電源や GND のラインは差し込まないようにして，作業を進めます．実際に電源を入れるのは，すべての配線が終わり，チェックが完了した後にしてください．

図 56 のように，ブレッドボード上部（**図 56** で黒くなっている位置）に LPC810 を差し込んだ状態は，**図 53** の状態の接続になっているので，この状態で起動させると LPC810 は ISP 書き込みモードで起動します．

第4章 応用製作編

製作事例について

LPC810を使った製作の例として，電子メール経由で外部機器のON/OFFを行うことができる，汎用のメール・リモコンと，あらかじめ登録した無線通信のモールス信号を送信する，ナノ・メモリ・キーヤの二つを作ってみましょう．

メール・リモコン

● メール・リモコンとは?

動作仕様

今回製作するメール・リモコンは，次のような仕様です．

- Windows PCに，USB-RS232Cアダプタで LPC810を接続しておく
- リモコン専用のメール・アドレス(フリー・メールで可)を介してやりとりをする
- 外部から，LPC810の4ポートのON/OFF，およびポート状態の取得ができる
- PCでのRS-232Cとメール送受信は，Windows PowerShellを使う

▶Windows PowerShell

製作に必要なハードウェアとしては，LPC810，USB-RS232Cアダプタ，電源用のパーツです．本書では，制御の対象は製作せず，GPIO経由で制御できる外付けの回路を各自で追加することで，汎用のリモート・コントローラとして運用することを想定しています．なお，本文では，サンプルとして4本のLEDの制御を行います．

ソフトウェアは，LPC Expressoで開発するLPC810用の制御プログラムと，パソコン上で動作させるWindows PowerShellスクリプトが必要です．これらのうち，Windows PowerShellは，Windows7以降では，OSに標準搭載されているため，インストール作業は必要ありません．Windows Vistaの場合，OSをSP1にしたうえで，.NET Framework 2.0以降をインストールし，32ビット版，もしくは64ビット版のWindows PowerShell2.0をインストールする必要があります．Windows XPでも，32ビット版ではWindows PowerShell2.0のインストールが可能ですが，Windows XPはサポートが終了しているため，ネットワークを利用するアプリケーションの常用はおすすめできません．

コントロール用のメールは，いたずらの可能性を減らすために，あらかじめ設定しておくチェック用の文字列を件名(Subject)に含んでいる場合のみ，本文に記載されているコントロール用のコマンドを実行するようにします．

コマンドはごく簡素なもので，ON，OFF，STATの三つです．それぞれ以下のようなパラメータをとり，メール本文に1行ずつ半角文字で，コマンド，対象ポートをスペースで区切って記述します．大文字，小文字は区別しません(**図260**)．

いたずら防止用のチェック文字列は，LPC810のコントロールに使うWindows PowerShellのスクリプト内で定義しておきます．たとえば，cq999，というチェック文字列を設定してある場合，**図261**のような動作イメージになります．

図261では，サンプルとして架空のメール・アドレスを使っています．

リモート・コントロール用メール・アドレス：

コマンド	対象ポート	本文の記述例
ON	0から3	ON 2
OFF	0から3	OFF 1
STAT	0から3(ダミー)	STAT 0

図260　コマンドの記述例

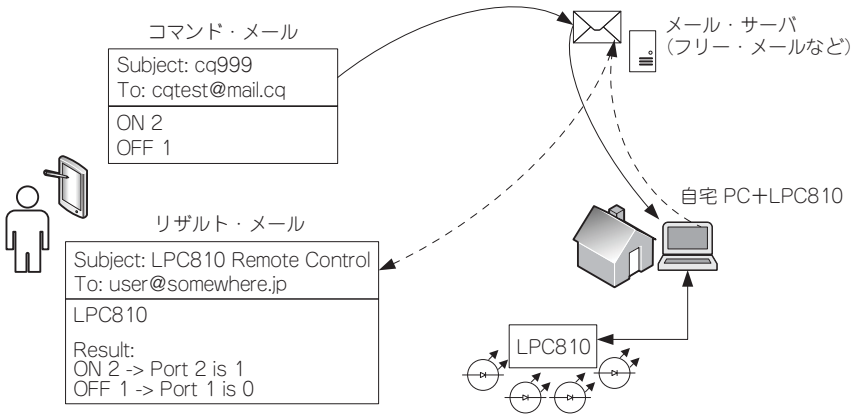


図 261
メール・リモコンの動作イメージ

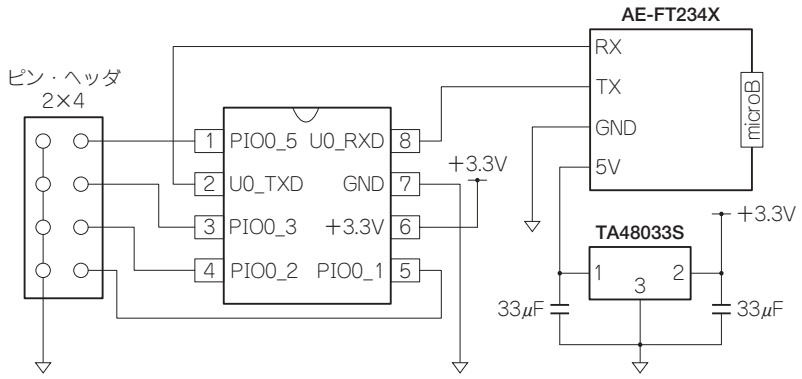


図 262
メール・リモコンの回路

```
  cqtest@mail.cq
ユーザのメール・アドレス：
  user@somewhere.jp
```

として，解説を進めます．手順は次のようになります．
※実際には，各自が使うことができるメール・アドレスを設定してください．

1. cqtest@mail.cq 宛てに，件名に“cq999”を含めてコマンド・メールを送信する
2. 自宅 PC の Windows Powershell が，定期的に POP でメールを受信する
3. コマンドを実行し，実行結果を user@somewhare.jp に返信する

コマンドとして ON や OFF を送った場合は対象ポートに対してコマンド実行を試みて，結果を返信します．STAT を送った場合，ポート指定は無視され，現在の四つのコントロール・ポートの状態を返してきます．ただし，STAT の場合も，ポート指定は省略できません．

STAT を送った場合の結果は，例えば次のようになります．

```
LPC810

Result:
STAT 1 -> R0:1 R1:0 R2:0 R3:0
```

これは，0 番のポートが ON，1 番から 3 番が OFF になっている，という状態です．

いずれの場合も，なんらかの事情で，LPC810 と通信できなかった場合は，タイムアウトのメッセージが返るようになっています．

ユニバーサル基板用回路

図 262 のような回路を組みます．PC とのインタフェースとして秋月電子通商の超小型 USB-232C アダプタ AE-FT234X を基板上に実装し，LPC810 とのシリアル通信と電源供給をまかないます．AE-FT234X からの電源は +5V のため，LPC810 へは，レギュレータを介して +3.3V の電源を供給します．

LPC810 で実際的な応用回路を組む場合，一点注意する必要があるのが，5 番ピンの ISP モード機能です．本文で解説したように，パッケージの 5 番ピンが

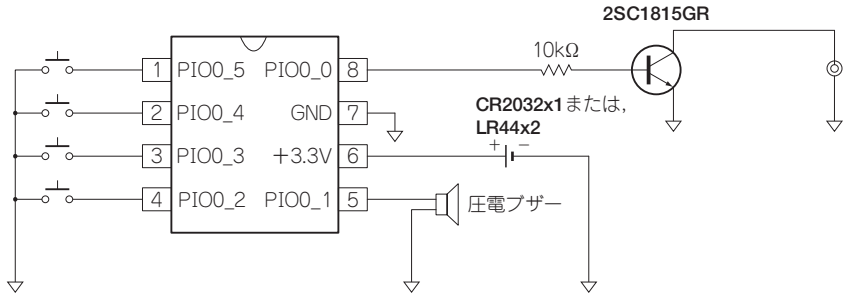


図 272
ナノ・キーヤのユニバーサル基板用回路

ヤを製作してみます．LPC810 の特長を活かし，できるだけ小型で簡素なものとするために，外部の拡張回路などは設けずに，LPC810 の 6 本の I/O 端子をフルに使った仕様で考えます．

全体の仕様は，以下のようなものとしています．

- 4 系統の送出符号列
- モニタ用圧電ブザー
- 送信符号列の書込・編集はプログラム書き込みで代用
- ボタン，またはコイン電池での動作

LPC810 の全 8PIN のうち，電源と GND を除いた 6 本のピンが GPIO として使用可能です．この 6 本を以下のように使っています．

GPIO0_0（ピン番号 8）……トランジスタ 2SC1815 経由でリグの KEY 端子へ
GPIO0_1（ピン番号 5）……圧電サウンダを使ったローカルモニタ
GPIO0_2 ～ 5（ピン番号 4 ～ 1）……送信系統選択スイッチ

送出符号列は，LPC810 のソース・コードにあらかじめ埋め込んで使うことにします．送出速度も同様にコードに埋め込んで，書き込み時に固定とします．LPC810 自体のポテンシャルとしては，スタンドアロンでの送出符号列の書き換えや送信速度可変も実装できないわけではありませんが，限られたピン数の中で，送出符号列の数を確保しつつ，それらを実現するとなると，外付けの回路が必要になります．そこまでやるのであれば，LPC シリーズにしろ，他のマイコンにしろ，もっとピン数の多いチップを使用したほうが素直に実現できるでしょう．

LPC810 の特長は，なんといっても小サイズ・低消費電力ということがまっさきに挙げられるため，ここでは，最低限のメモリ・キーヤとしての機能に絞り，ナノ・サイズで電池駆動できるという実装を考えてみます．動作時の電流は，実測で約 2.3mA ほどで，CR2032 の場合は，連続負荷としては，最大 3mA とはいえ，ちょっと苦しいかというところですが，それでも 170mAh 程度の容量はありそうで，単純計算では，73 時間くらい動作できることになります．

ユニバーサル基板用回路

ナノ・キーヤのユニバーサル基板用回路は，**図 272** のようになります．電池に関しては，電池ホルダの規格がまちまちなため，別途ユニバーサル基板を各自で用意していただき，ユニバーサル基板上のピン・ヘッダとユニバーサル基板上のピン・ソケットで連結する仕様としています．

前述したように，CR2032 での駆動は，電池の規格上ぎりぎりなところもあるため，電池駆動に不安がある場合には，たとえば単 3 や単 4 などの乾電池にしたり，ステップアップ・コンバータでの昇圧，もしくは 3.3V 以上を用意してレギュレータでの降圧にしたりすることも可能です．

なお，仕様検討の過程では，秋月電子から出ているステップアップ・コンバータ HT7733A（秋月電子通販コード M-05720）と，CR2032 の組合せでも動作できていますが，できるだけ簡素化するという方針から，あえて電源周りの回路はなしの構成としています．

また，ISP モード問題のある 5 番ピンは，圧電ブザーが接続されていて，テストした環境では起動時に GND 判定されることはありませんでしたが，問題が出るようであれば，タクト・スイッチのいずれかに 5 番ピンの接続を変更し，プログラムを修正してみてください．

● ナノ・キーヤのハードウェア

パーツ・リスト

ナノ・キーヤのパーツ・リストは，**図 273** のようになります．圧電サウンダやタクト・スイッチは，指定型番以外のものでは，ピンのピッチが合わない可能性もあります．

モノラル・ジャックの接続は，スリーブ側が接地と

おわりに

本書では，省電力，小サイズ，高速I/O，3種類の多機能タイマ，ROM上のI^2CとUARTを持つ8ピンDIPマイコンLPC810について，その魅力を楽しむための事例を中心に紹介してきました．

4ビットの4004からスタートしたマイコンも，今では8ピンのサイズの中に高機能な32ビットCPUコアが入った製品が100円を切る価格で簡単に手に入ってしまう時代となり，多彩な応用が可能になった反面，はじめてマイコンに触れるユーザにとっては開発環境の敷居が高かったり，従来のマイコン環境になじんできたユーザにとっては，とりあえず使えはするもののマイコンを使っているという感覚が以前に比べて薄れていると感じたり，という側面もみられるようになってきているのではないでしょうか．

LPC810は，筆者にとって久しぶりにマイコンらしいマイコンを扱っているという感覚を思い出させてくれるものでした．記憶領域も，I/Oピン数も，十分にあるマイコンでは，統合開発環境内で抽象化されたAPIのライブラリ群をコールすることで，マイコンの内部事情にあまり煩わされることなく，やりたいことが実現できるような世の中になってきています．それはそれで，ホビーや勉強の用途からも良いことですし，産業への応用面でも，生産性や保守性の観点から必然的な流れであると思います．

しかし，逆に制約の多いLPC810では，やれることはなにか，やりたいことを実現するために，ほんとうに必要なことはなにか，ということをよく考えなければならない場面にしばしば遭遇します．本文でもみてきたように，機能自体は高いものを持っているLPC810ですが，それを活かすには，ユーザの創意工夫が求められるという点で，LPC810はマイコンと戯れるには最適な選択肢の一つではないかと感じています．

本書では，LPC810でARMマイコンに入門してみたいという方や，LPCシリーズに興味があるけれど，開発環境や資料の掘り下げに手間がかかりそうだと感じている方を念頭に，何かをするために必要なことだけに絞って，思い切った簡略化をした上で，スタートアップに役立ちそうな話題を集めて紹介してみました．

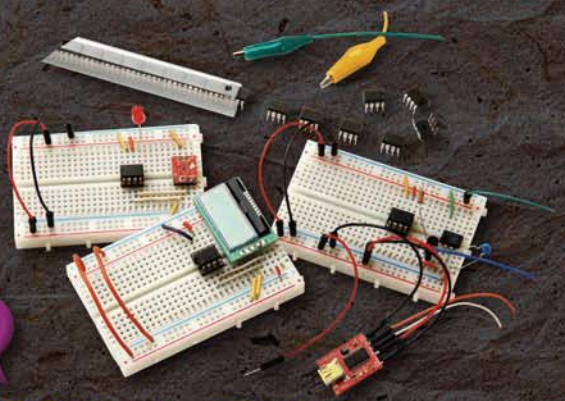
LPC810の機能のうち，SPIは必要なピン数が多くなること，ADCMPを使用したソフトウェアA-Dは，コード・サイズを圧迫する，もしくは割り込み系統を占有するという観点から，あえて触れない選択をしました．Watchdogタイマのようなフェール・セーフ，省電力化のための，Power ManagementやWake Upタイマの機能についても取り上げるかどうかは悩みましたが，本書の性格上，そこまでの応用は他書に譲るべきと考えて省略しました．

本書を通して，LPC810の基本的な読み解き方をつかんでいただければ，やがては自力でこれらの項目についても開拓していっていただけるのではないかと期待しています．

最後になりましたが，辛抱強く待ち続けてくださったCQ出版社編集の今　一義さん，執筆を支えてくれた妻と黒猫に感謝の意を表してまとめとしたいと思います．

本書が，LPC810でマイコンを楽しんでみたい方のなにかの手がかりになれば，筆者としてそれ以上の喜びはありません．

2015年3月　中村 文隆



挿すだけ！
ARM32ビット・マイコンのはじめ方