オブジェクト指向マイコンプログラミング

伊集院 八郎

PC 上ではすでに構築済みのオブジェクト指向プログラミング環境が、プログラミング言語を C から C++へ上げることによりマイコン上でもやっと構築できるようになった。その環境にクラスライブラリを含めることにより、PC とマイコンの間の通信が効率よくなされるようになった。そのライブラリにある Serial クラスやそのインスタンスは、マイコンの I/O の出力へ PC からの制御パラメータ設定のためのメッセージを送ることができ、またマイコンから逆に送られたメッセージは PC のディスプレイ上に描かれたものに影響を及ぼすようになる。この環境に関する展望について二三深く論じる。

Object_oriented Programming for Microcontrollers

Hachiro Ijuin

Object_oriented programming environments for some kinds of microcontrollers have been established recently through raising the programming language C up to C++, while they were already done on the PC. Inclusion of class libraries into the environments has provided effective tools for communication between the PC and the microcomputer. Serial class in the libraries and its instances can send messages from the PC to the outputs of microcontoller's I/O to set proper parameters for control and the messages sent reversely from the microcontroller can affect the objects drawn on the display of the PC. Some other prospects concerning to the environments are discussed further .

１．はじめに

マイコン（マイクロコントローラ）は、数値の計算やデータの処理よりは、周辺装置のコントロールを得意とし、そのための機能を充実させてきた。それと同時に、周りに置かれた装置の機能を次々に取り込んで、ワンチップ化することも図られてきた。一方、ソフトウェア面は、メモリの利用効率やアクセスの速さの点から、初めは機械語に近いプログラムが好まれ、アセンブリ言語プログラミングがしばらく続いた。しかしながら、プログラムの可読性を改善して開発効率を上げるためには、レジスタや入出力を徐々に抽象化していく必要が生じ、マイコンの開発言語も次第に低級言語から高級言語へと推移していった。つまり、高級言語でありながらポインタ操作のできる C 言語が次の段階のマイコン開発言語として登場し、今なおその中心に置かれている。

しかしながら、今やマイコンはその動作周波数がかなり上がって高速になり、搭載メモリも増大しつつあることから、OS を初めとする大きなプログラムをも実行できるようになってきた。近年のソフトウェア開発効率は、オブジェクト指向のプログラミングによって著しくたかめられたことを考えれば、マイコンのプログラミングにもその流れが浸透してくることは必定と言えよう。

ここでは、このマイコンのオブジェクト指向プログラミングについて、これまでと現状を見つめながらその導入のされ方やマイコンプログラミング特有の利用状況について述べ、これからの発展の方向性について議論する。

==========

近畿大学　産業理工学部
Kinki Univ.

２．マイコンのプログラミング環境

マイコンの基本構成は、CPU に RAM や ROM のメモリを備えたものから、I/O ポート、USB などの通信ポート、周辺機器を接続するための各種ポートあるいは A/D や D/A の変換機能を含めたものなどへと発展し、オールインワンのチップ構成のものが普通になってきた。その主な使用形態も、電子機器や装置を高性能化するために単独で使用されるものから、複数個組み込んで使用されるものまで多岐に亘ってきている。

しかしながら元来、マイコンの搭載は周辺の機器や装置の制御が目的であり、しかも単純な機能追加に限定される場合が多かったため、複雑な演算処理や大量のデータ処理には向かない構成、つまり、クロック周波数はそれほど高くなく、搭載メモリも少な目であった。また、制御のためには、センサ入力を処理して、その結果によってアクチュエータを駆動することが基本になるので、信号の入出力が頻繁に行われ、それに応じてレジスタの値を更新することが主な動作であった。このようなレジスタ操作をプログラムするのにもっとも適しているのは機械語であるため、マイコンのプログラミングは機械語から始まった。続くアセンブリ言語は人間に分かりやすい言語に近づきながら、まだ機械語とは一対一に対応するものであった。　一方、アセンブリ言語のプログラムが複雑になり、高級言語による開発も始まると、もはやマイコン上でのプログラム開発には限界が生じ困難となっていった。このように機械語のプログラミングから離れるに従い、マイコンのプログラミングはそのメモリ上ではなく、他のコンピュータ上でプログラミングを行ない、そのソースコードをクロスアセンブラやコンパイラによってマイコンで実行可能な機械語のプログラムに変換する方法に変わっていった。そのため、プログラムの開発環境とは別に、機械語のプログラムをマイコンへ書き込むためのハードとソフトが新たに必要になった。この場合、書き込み時にマイコンはそれを搭載する実機からは切り離さなければならなかった。しかし、その後マイコンを実機から切り離すことなく、埋め込んだまま書き込みを行えるインライン書き込みも行えるようになった。

一方、このインライン書き込みは、これまで長くパソコン間通信に多用されていたシリアル通信を利用するものであったが、USB 機器の普及と相まって、パソコンとの USB 接続によるものが増えてきている。この USB 接続は、マイコンに必用な電源を供給するためにも利用できるようになっている。さらに最近では、マイコンをパソコンに USB 接続し、Web 上で編集したソースコードをリモートサーバ側でコンパイルし、併せてマイコンへの書き込みもできるようにした開発環境も現れてきた。

3. マイコンのオブジェクト指向プログラミング

論理演算を行うゲート回路やメモリを構成する集積回路としてのロジック IC やメモリ IC などの単体とは異なって、プログラムによって素子接続の再構成が可能ないわゆるプログラマブル IC の出現は、ハードとソフトをはっきりと二分することを難しくしている。MPU(Micro Processing Unit) や DSP(Digital Signal Processor)のような高度な演算を処理するプロセッサ IC に対し、特定用途のものとしてカスタム化して使用する目的の集積回路 ASIC(Application Specific Integrated Circuit)が開発された。続いて、そのカスタム性の自由度をさらに高めることにより、製造後にも再構成が可能な FPGA(Field Programmable Gate Array)が生産されるようになった。開発のリスクをおさえ、さらに開発の期間を著しく短縮可能にしたのはこのタイプで、そのプログラミングスタイルはこれまでの C 言語ベースのものから C++ベースのものへ変わり、さらに Java ライクのものに変わりつつある。プログラマブル IC としては、他に CPLD(Complex Programmable Logic Device)もあるが、これは FPGA とは異なってプログラミング素子を不揮発性にし、集積度をやや低くしたものが多い。これらのプログラマブル IC の特徴は、ハードウェアが機能を固定させて回路の無駄を無くし信頼性や高速性を高めることのできる利点と、処理手順を柔軟に変えて目標の機能を実現しやすくできるソフトウェアの利点とを併せ持っていることである。このソフトウェアの拡張性や開発効率の高さを十分に活かすために、必然的にオブジェクト指向のプログラミングが MPU や DSP を含めたマイコンの世界に浸透してきたと言える。

4. フィジカルコンピューティング用のツール群

昨今、フィジカルコンピューティングの話題が多く取り上げられるようになってきた[1]。すでに一部マニアの趣味の域を脱して、広い年齢層に亘って創作意欲を駆り立ててくれる好例到来の感さえ漂わせている。その魅力のもとを手繰ると、ソフトウェア開発におけるオブジェクト指向の恩恵を随所に垣間見ることができる。しかしながら、フィジカルコンピューティングを可能にするハードウェア面については、何がどれだけ必要で、どれほどの準備で足りるのかといった議論は必ずしも十分ではない。それゆえ、フィジカルプログラミングに言及する前に、そのツール群の構成の全体像をまず明示することにする。

フィジカルコンピューティングのためのツールの中核にはマイコンが置かれる。そのマイコンは、入出力信号に応じてレジスタの値を更新するのみであると極論すれば、マイコンの外には信号の送り手と受け手を用意するだけで良いことになる。そのうち、そのいずれも不要な場合であれば、内部のクロック信号を利用するか内部でデータ処理するかして内臓の不揮発性メモリに記録を残すことなどが考えられる。しかし、外界との接触のない処理ではフィジカルコンピューティングの部類に含めることはできない。それゆえ、少なくとも入出力の一方の存在は不可欠である。

まず、注目されるのは出力用ツールである。現実には、光や音あるいは物の動きなどがマイコン出力の代表例であるが、実際にマイコンの出力端子に現れるのは規定の電気信号に過ぎないので、最大レベル以下の電圧測定可能なメータの針の変位や動きが出力の全てと言える。しかしながら、そのようなメータの針の動きなどに対しては、ほとんどの人が関心を持つことはない。どうしても他の物理量が望まれる。マイコンの出力側に置かれるもっともポピュラーなものには、発光するダイオード LED(Light Emitting Diode)が普通は選ばれる。消費電力が少ないだけでなく、安価で壊れにくく入手しやすいのがその手軽な利用に結びついている。しかし、ダイオードの最大の特徴である長い足から短い足の方へしか電流が通らないという流れの一方向性、発光ダイオードであれば 1.5 ボルトの乾電池１本ではほとんど光らないが、2 ボルト付近からは急激に電流が流れ出すので乾電池 2 本（3 ボルト）では焼き切れてしまうこと、それゆえこの場合は必ず電流制限用の電気抵抗（５ボルトまではおよそ 250 オーム以上で上限は明るさ次第）を直列につなぐことだけには十分に留意しなければならない。このような LED の性質を考えても、その点灯や消灯によって信号のハイレベルとロウレベルを検出するディジタル出力用として使用するのに関しては全く問題はない。しかしながら、LED をアナログ出力側に置く場合は注意しなければならない。つまり、LED は 2 ボルト未満になると急に電流が流なくなり、出力が最低レベルや最高レベルに達する前に消灯してしまうことを十分に承知の上で使用しなければならない。図１の中央の上下二つの回路が、ディジタル出力にもアナログ出力にも使おうとする LED の回路を示している。それぞれハイレベルおよびロウレベルの出力に対して点灯する。アナログ出力に対しては、上下それぞれの LED が出力の最低レベル前および最高レベル前で消灯する。

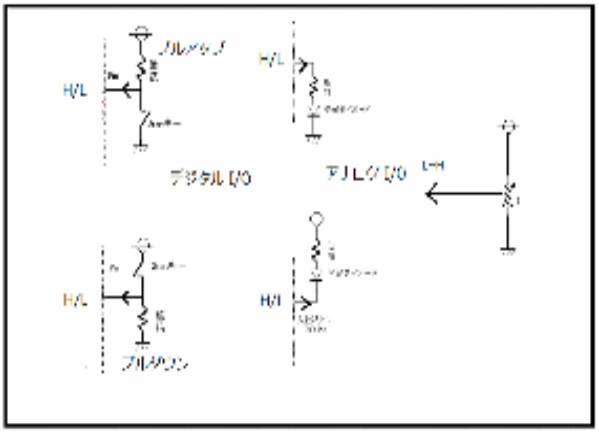


図１　マイコンのアナログ/ディジタル入出力

他方、入力ツールになるものとしては、ディジタル信号を扱う場合とアナログ信号を扱う場合を分け、それぞれ図１の左のスイッチと右の可変抵抗を入力側の代表として充てることができる。ただし、図の左のスイッチは、その上のものがスイッチオンでロウレベルの入力をマイコン側へ伝え、下のものがハイレベルの入力を伝えるものになる。結局、フィジ

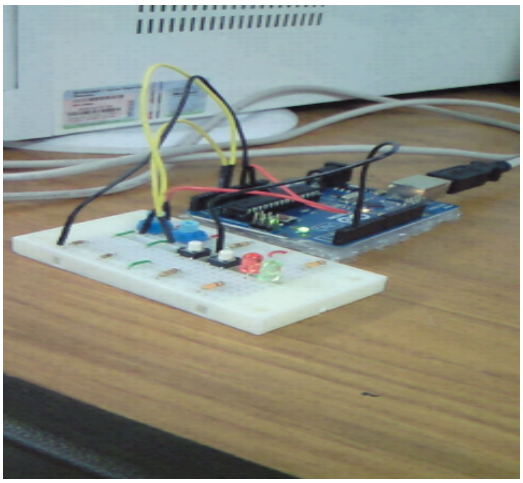
カルコンピューティングのための最小構成のツール群としては、マイコン以外には出力用の LED と入力用のスイッチおよび可変抵抗で足りることになる。どのように複雑な電気回路を組んでも、マイコンの周りはこの３種が形を変えて置き変わったものと見做すことができる。

5. Serial クラスライブラリの応用

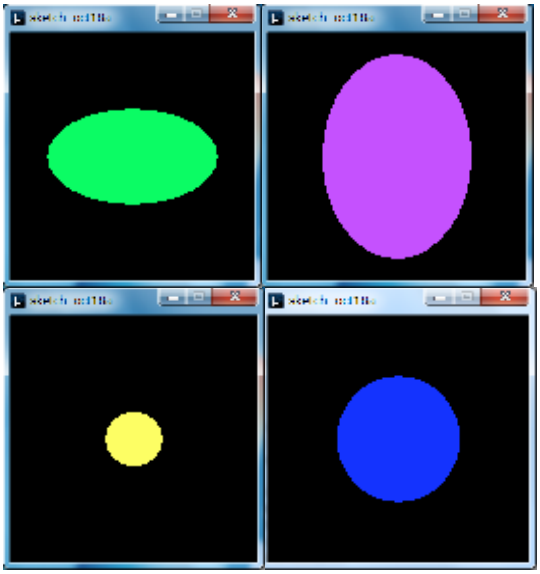
フィジカルコンピューティングでは、その開発環境の著しい質的向上が間違いなく大きな魅力となっている。さらに、オブジェクト指向プログラミングが浸透し、開発効率を大きく向上させたことが快適なプログラミングを可能にしている。ここではオブジェクト指向プログラミングによる恩恵が、クラスライブラリを利用するときに得られる例に留める。

AVR マイコンにアナログやディジタルの入出力デバイスを手軽に接続できるようにし、しかもその開発環境までを提供する Arduino は、フィジカルコンピューティングに適したマイコンモジュールである[2]。パソコン側の開発環境下で C 言語風のプログラミング言語によってコーディングすれば、そのコンパイルからマイコンへの書き込みまでを画面のメニュー項目やアイコンの選択だけで簡単に続行できる。ただし、電源供給と通信のために、USB ケーブルをあらかじめ接続し、その通信ポートを正しく設定しておかなければならない。Arduino では、そのプログラム（スケッチ）をコンパイルしてマイコンに書き込むと（アップロード）、同時にシリアルクラス Serial のクラスライブラリが組み込まれる。それゆえ、これによって、マウスクリックだけで立ち上げることのできるパソコン側のモニタ画面とマイコンとの間での通信がいつでも可能になる。

また、そのサンプルの中には他言語とシリアル通信を可能にするためのものも用意されている。そのソフトを立ち上げることによって、Java の文法に沿ってプログラミングできる言語 Processing で作成したアプリケーションとマイコンを連携させられるようになる[3]。元来、この言語は、必要最小限のプログラミングコードによって描画が可能なことから、デザイン指向のプログラミングに適している。ここでは、図 2 に示されるように、上段の図のマイコン側のアナログ入力端およびディジタル入力端から USB ケーブルを通して送られる描画用のパラメータによって、下段の図のパソコン側の画面上に楕円を描き、その位置、形および色を任意に変えられるようにしている。



（上段） マイコンとその周辺回路



（下段）パソコン画面への描画

図 2 マイコンからの指示によるパソコン画面への描画

また、これとは逆に、パソコン画面からの指示によって、マイコン側の出力端をコントロールすることもできる。例えば図 3 に示されるように、左上側の図の右半面にマウスカーソルがあるときには画面に変化はなく、マイコン側の LED は消灯しているが、

マウスカーソルを左半面に置くと、左下側の図のように画面の色が変化してマイコン側の LED が点灯する。また、図の右半面にマウスカーソルを持ってくると、左端から右端へその位置を変えることにより、マイコン側の LED の明るさを真っ暗から最高の明るさまで連続的に変化させることができる。

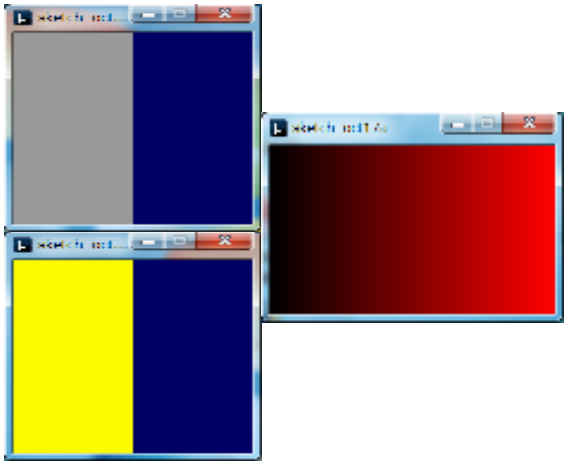


図 3　マイコン側の LED の点灯と消灯および
明るさを変えるためのパソコン側の画面

データに変える adc オブジェクトの出力をメトロノームのリズムごとにスナップショット抽出し、そのデータによってマイコンの LED への出力を PWM(Pulse Width Modulation)制御して明るさ調整するものになっている。

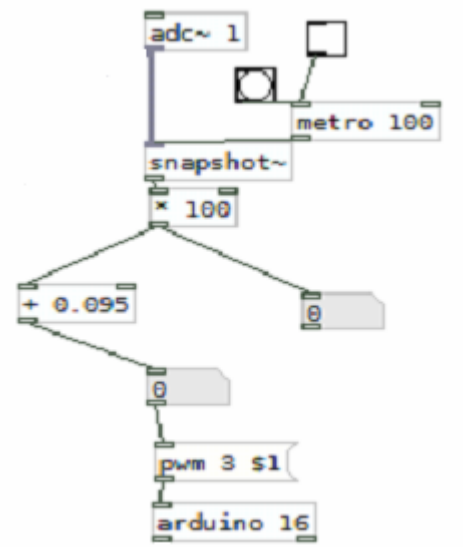


図 4　オーディオ入力を LED の明るさの変化
に変える Pure Data プログラム

これらの描画や入出力制御の例では、パソコンとマイコンの間の通信の設定やコントロールを双方の同名のクラス Serial やそのインスタンスが担っている。Processing では、マウスイベントも簡潔なプログラム記述によって処理できるようになっている。Processing に関しては、そのコードを HTML ファイルのスクリプトタグに埋め込んで、Web プログラミングで利用できるようにした Processingjs も開発されている。

一方、さらにマイコンとその周辺装置をも抽象化し、パソコン側でそれらの代理となるクラスやインスタンスにパソコン側の処理を担わせるプログラミングスタイルをとることも可能になる。例えば，グラフィカル言語の Pure Data (pd)を用いれば、この言語用のマイコン Arduino のクラスライブラリが開発されているので、これを利用するプログラミングが可能になる[4]。この言語は、音声の処理を得意とする言語ゆえ，図 4 ではオーディオ入力に対し、マイコン側の LED の明るさを変調させる例を示している。基本的な処理の流れは、オーディオ入力を音声

クラウドコンピューティングの波は、マイコン開発に対しても新しい流れとして押し寄せ始めている。マイコンボード mbed は、ARM マイコン CortexM-3 シリーズの MPU を搭載したプロトタイピング用ボードと位置付けられている[5]。開発環境をインストールすることなく Web 上での開発が可能で、ボード以外にはパソコンとの通信用の USB ケーブルおよびブラウジングソフトのみが必要になる。どの OS であれ、そのブラウザが起動してインターネット接続できさえすればよい。C++言語によるプログラミングが可能で、基本的なヘッダーファイルの取り込みによって、入出力クラスは不自由なく使えるようになる。例えば，アナログ入力とディジタル出力のクラスを利用し、入力信号によって出力の状態を変えるプログラムは、次のように簡潔なものになる。つまり、それぞれのクラス AnalogIn および DigitalOut のインスタンスを入出力ピン 20 番および 21 番などを指定して ain(p20)や dout(p21) のように生成し、ain の真偽に応じて dout にハイレベルやロウレベルの出力値 (1 や 0) を代入する処理を無限ループ内に置く

だけでよい。図 5 に示されるようなピンの割付は、ヘルプ画面などを利用して即座に確認することができる。

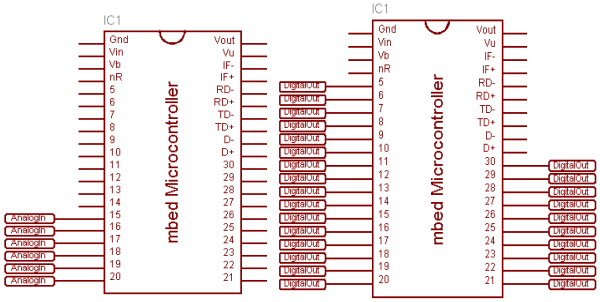


図 5　mbed プログラミングで必要になるアナログピン（左図）とディジタルピン（右図）の配置

6. マイコンプログラミングの展望

マイコンのプログラミング環境は急速に大きく変貌を遂げようとしている。高精度の制御は当初より望まれることではあったが、なかでも実時間制御への要請が増大しつつある。また、マイクロコンピュータが、その演算速度と搭載メモリの値を次々と刷新していったように、マイコンが瞬く間に互いの隙間を埋めていくことも考えられる。これまでは、一部のマイコンを除いてその外で考えられていた信号の処理などの重い処理は、次第に高機能マイコン側へ吸収されていく勢いも感じられる。

柔軟なシステム開発に適した FPGA は、ディジタル回路の書き換えが主であった。今ではアナログ回路の書き換えの可能なものも現れている。一方、PSoC(Programmable System on Chip)は、はじめからディジタル回路とアナログ回路の両方の書き換えが可能なマイコンとして登場した。FPGA のディジタル回路をプログラムする能力と、アナログ回路として有用な増幅や発信あるいはフィルタの回路などをプログラムできる能力の双方を合わせ持つ極めて高機能で高集積度のマイコンが出現したことになる。このような新しいマイコンに対しては、ソフトウェア技術もますます高度なもので対処することが要求されるであろうから、一般のコンピュータに適用されるのと変わらない普遍的な扱いが、これからのマイコンにはなされていくものと考えられる。

7. まとめ

マイコンのプログラミングのこれまでと今を概観し、その機能強化とともに、プログラミングスタイルも大きく変貌してきていることを示した。ソフトウェア開発の常道としてのオブジェクト指向プログラミングは、ここでもその主流となりつつあることをみた。プログラムは無理なく開発できるパソコン側で行って結果をマイコン側へ渡すのにも、複雑な処理をパソコン側で行ってマイコンと連携して動作できるようにするためにも、シリアル通信に重要な役目を担わせた。さらに、ネットワークを介した通信に広げれば、ネットでつながるマイコンやパソコンの連携動作ができ、リモートコントロールや新たな分散処理の形態が生み出される可能性もある。

質疑応答

Q （小出）マイコンのレジスタに触れないプログラミングとその教育実践に関して.

A マイコンプログラミングの基本はやはりレジスタの動きを正しく記述すること。しかし、通信などを考えるとそのためのクラスライブラリが扱えるオブジェクト指向言語を使用した方が開発効率は格段に高まる。担当する実験テーマ「組込みコンピュータ」では、PIC マイコンのアセンブリ言語プログラミングに C 言語プログラミングを加えている。今後は、抽象化の高い言語も必要に応じて加えていく。

Q （山之上）Processing 言語の使いやすさについてと組み込みシステム研究会案内

A Processing 言語は描画に優れた言語で大変使いやすい。描画用の関数を記述するだけでよい。Java の文法に準拠していながら、Java（アプレット）のような前後の記述は不要。あいにく、学会が重なって組み込みシステム研究会へは不参加。

[参考文献]

[1] 情報処理学会誌, Vol. 52, No. 8 (2011) 特集ほか
[2] http://www.arduino.cc/
[3] http://processing.org/
[4] http://at.or.at/hans/pd/objects.html
[5] http://mbed.org/