

小ピンマイコン付DIP変換ボード 「FB-78F9222MC」を利用した電子ピアノ

神奈川県
戸井田 貞弘 様

はじめに

電子ピアノを小ピンマイコンで

今回は8ビット・マイコン「 μ PD98F9222」を搭載したDIP変換ボード「FB-78F9222MC」を使った単音出力の電子ピアノを紹介します。概観は写真1で、液晶ディスプレイには動作モードを表示します。

単音出力で同時押しには対応していませんので、和音などは扱えませんが2オクターブ強の音階に対応しており、更に録音／再生機能も持たせています。ちょっとしたメロディラインを追ってみたい時や、子供向けの玩具としては十分楽しめるでしょう。

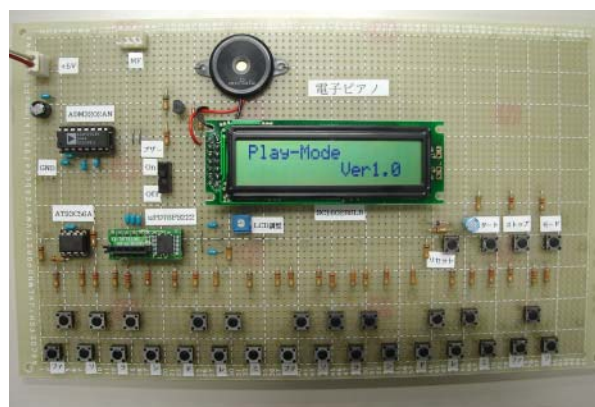


写真1

電子ピアノの概要

今回紹介する電子ピアノの構成は図1のようになっています。基本機能としては

- 27鍵(2オクターブ+3音)のキー入力
- 液晶ディスプレイへの動作モード表示
- 外部EEPROM接続による63音までの録音／再生機能
- シリアルポート経由でのリアルタイムメモリリード／ライト機能(メモリファインダー)

の四つが大きな柱となっています。動作モードは次の三つで、モード切替スイッチで切り替えます。

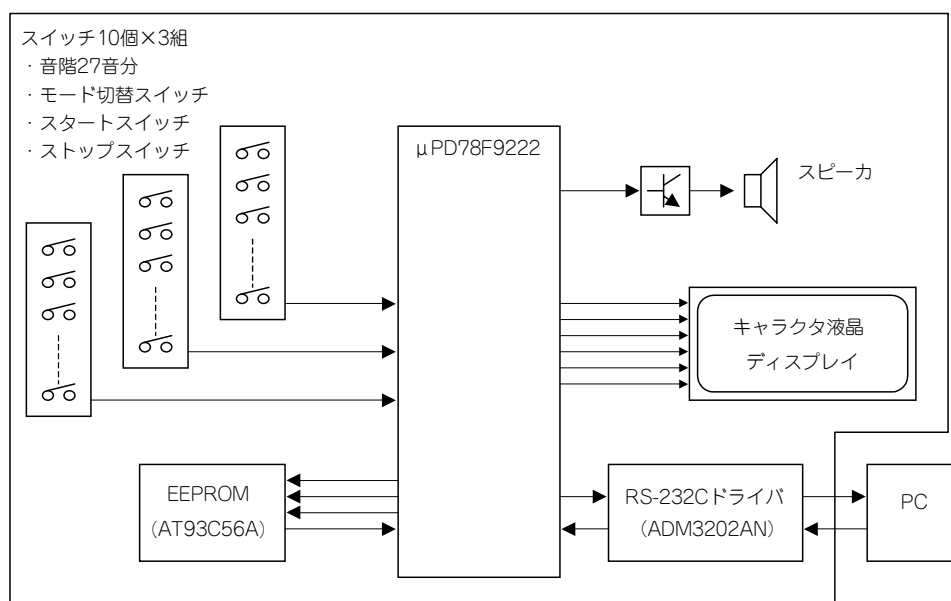


図1 電子ピアノのブロック図

- (1)演奏(PLAY)モード
- (2)録音(RECORD)モード
- (3)再生(REPLAY)モード

次にそれぞれのモードの動作について説明していきましょう。

●演奏(PLAY)モード

通常の楽器として利用するのが(1)の演奏モードです。FB-78F9222MCに内蔵されている16ビットタイマであるタイマ00出力(TO00)を使って、指定された音階に対応する周波数を出力させます。音階の周波数はいわゆる平均律で1オクターブの間を12個の等比数列で埋めます。440Hz(ラの音)を基準にしたので、 $440 \times (2^{(n+12)})$ (nはラの位置からいくつめのキーかを示す)の周波数になるように、カウンタの値を計算しておきます。1オクターブ上のラならば、 $440 \times (2^{(12+12)})=440 \times 2=880\text{Hz}$ ですし、ドならば、ラ#、シ、ドと、三つ上のキーになりますので、 $440 \times (2^{(3+12)}) \div 523.3\text{Hz}$ になります。

タイマのクロックは8MHzですのでたとえば440Hzを出力するならば、カウント値は $8 \times 10^6 \div 440 \div 18181.818 \dots$ となりますので、18182をセットするように思えますが、FB-78F9222MCのタイマを方形波出力モードで使ったときは、カウントアップした時点で出力が反転するという仕様です。したがって、この値をセットすると約2.273ms ($1 \div (8 \times 10^6 \div 18182)$ 秒)ごとに出力が反転しますので、周期は二倍の4.546ms、つまり220Hzになります。

440Hzを出力したければカウント値を半分の9091にします。このようにして計算した結果がソースコードのQ_MELラベルのところのテーブルです。

なお今回の電子ピアノでは元となる8MHzのクロックにはマイコン内蔵のリングオシレータを用いていますので、水晶発振器などと比べると誤差や温度による変動が大きめです。またカウンタの端数の影響などもあり、出力周波数には多少の誤差が生じます。

より正確を期すならば、水晶発振器を用いカウント値による誤差が少ない周波数を選ぶなどの工夫をするのが良いと思います。

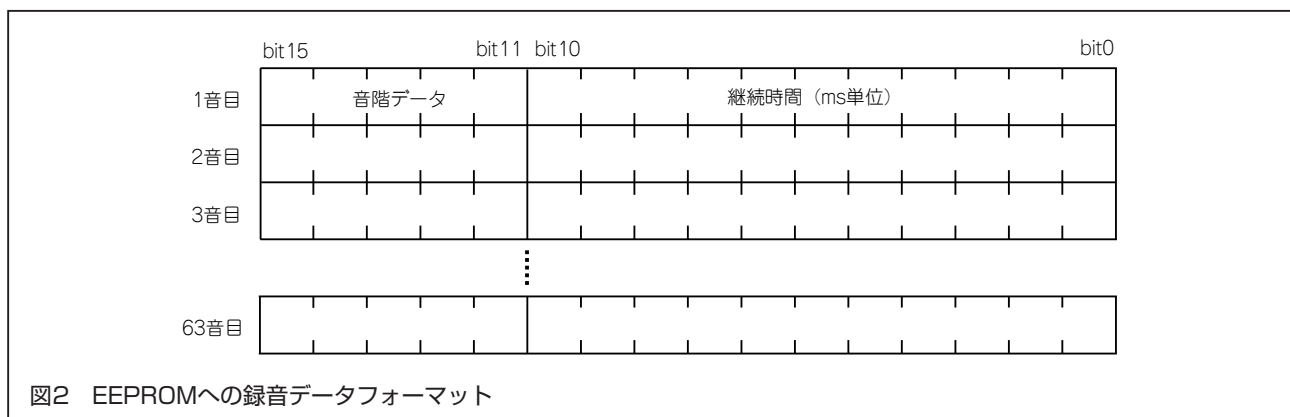
●録音(RECORD)モード

録音モードで、スタートスイッチを押すと、以後の演奏状態を記録していきます。ストップスイッチを押すと記録動作を中止します。記録できる音の数は63個ですので、ストップスイッチを押さなくても63個までいくと自動的に録音モードを終了します。

データの記憶は外部に接続したEEPROM(AT93C56A)を利用し、押下されたキーのコードと、押下継続時間を書き込みます。EEPROMは不揮発性のため電源を切っても記憶内容は消えません。一回録音したものはいつでも再生することができます。

EEPROMはフラッシュメモリのようにブロック単位やチップ単位で丸ごと消去しなくてはならないものと異なり、バイト(ないしワード)単位での書き換えが行えますので扱いが簡単なのも利点です。

EEPROMに格納するデータは図2のように、1音あたり16ビット長で、上位5ビットがキーコード、下位11ビットが押下時間(ms単位)です。押下時間データが11ビット分ありますので、最大値は約2秒(2047ms)となります。2秒以上継続した場合には2秒として記録します。

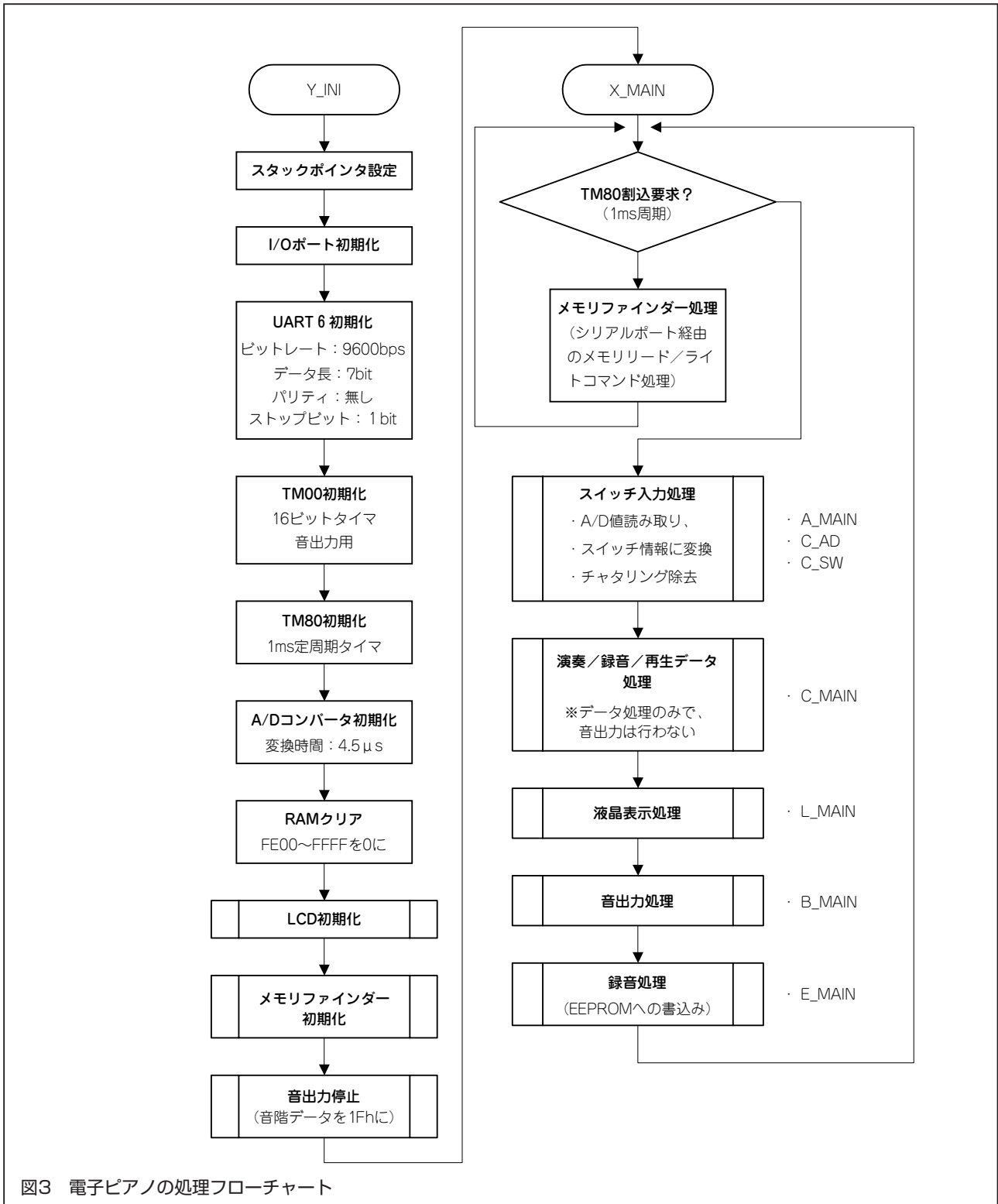


●再生(REPLAY)モード

再生モードでスタートスイッチを押すと、EEPROMに格納された音楽データを再生します。再生中にストップスイッチが押されると再生を中断します。

●電子ピアノの動作フロー

電子ピアノ全体の動作フローチャートは図3、RAMのメモリマップは図4のようになっています。フローチャートブロックの横に書いた名称はソースコード中のラベル名ですので、ソースコードとつき合わせるときの参考にしてください。



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
FE00	ZXBSTCK															
FE10																▲
FE20	ZRPBUF 送受信ポイント	▲ ZRPBUF 送受信バッファ														
FE30																
FE40							▲ ZABANI0 ANI0今回の値		ZABANI1 ANI1今回の値	ZABANI2 ANI2今回の値	ZCBANI0 SW番号(00-09)	ZCBANI1 SW番号(00-09)	ZCBANI2 SW番号(00-09)	ZCBSWN 今回SW番号	ZCBSWZ 前回SW番号	ZCBSWU 立ち上がりSW番号
FE50	ZCBSWK 確定SW番号	ZCBSWZK 前回確定値	ZCSCW 30回カウンタ	ZLSSTG 表示ステージ	ZLBDISP 表示データエリア (16文字×2行)											
FE60																
FE70				▲ ZLDPDISP 表示データポイント	ZLDPDISP 表示データポイント	ZCSMODE メイン機能番号	ZCSREP 再生モード	ZCSREC 録音モード	ZCCMEL 鳴動時間カウンタ			ZCBSWW 前回押下SW (C_REC)				
FE80	ZEBBUF データバッファ (最大8ワード)															
FE90	ZEBDATA 書き込みデータ	▲ ZESSTG EEPROM 状態番号	ZESSTG EEPROM 状態番号	ZEBADRS EEPROMアド レス(00-3F)	ZECWDT ライトデータ数 (未書きデータ=0)	ZECWAIT 書き込みタイムアウト 検出カウンタ(12ms)	ZBBMEL 音階番号指定 (00-1A,1F)									
FEA0																
FEB0	ZOBDBG デバッグワーク															
FEC0																
FED0																
FEEO																▲
FEF0	ZCPDBG ポイント	▲														

図4 FE00 : RAMマップ

スタックの初期設定やタイマやLCDの初期化などを行った後、X_MAINルーチンにジャンプします。ここではタイマ80の割り込み要求フラグをチェックしています。タイマ80は1ms周期で割り込みを発生するようになっています。

今回、割り込みは使わず、タイマ80の割り込みステータスをポーリングして処理しています。タイマ要求フラグが立っているかをステータスレジスタを読み出して確認し、立っていればビットクリア命令でクリアして、ピアノ関係の処理を行います。タイマ割り込みが発生するまでの間はピアノ関係の処理は必要ありませんので、メモリファインダーの処理を行わせました。

タイマ80割り込み要求の処理では、まずスイッチを読み込みます。後で説明しますがスイッチはA/Dコンバータで読み取るという、少し変わった方法をとっています。読み取ったデータからスイッチ情報に変換します。更に30回(30ms)連続して安定して読めたことで正式な変化として採用でき、ON/OFF時の接点の暴れ(チャタリング)の影響を除去しています。

続いて、演奏／録音／再生処理を行います。ただし、ここでは実際の音出力などは行いません。後の音出力サブルーチン等のためのデータ処理を行うだけです。モードの変更や液晶ディスプレイへの表示データのセットアップなどはここで行います

この処理の後は液晶ディスプレイへの実際の表示処理、タイマへの出力処理、録音モード時のEEPROMへの書き込み処理などを行ってX_MAINに戻ります。

ソースコードを読むときは、先ほど触れたとおりデータ処理部分と実際の出力処理部分が分けられていることに注意してください。

電子ピアノの主要ブロックとその動作

今回製作した電子ピアノの内部は次の4つの機能ブロックに分けて考えることができます。

- キー入力機能
- シリアルEEPROMのリード／ライト機能
- 液晶ディスプレイ制御機能
- 音出力

音出力はタイマの出力周波数変更だけでするので比較的簡単ですが、その他の3つについては少々複雑ですので、ここでこれらの機能ブロックの動作について説明しておきます。

● キー入力の仕組み

今回のようにピン数の少ないマイコンで多数のスイッチの読み取りが必要な場合、入力回路にも一工夫必要となります。ここでマイコンのスイッチ入力方式について考えていきましょう。

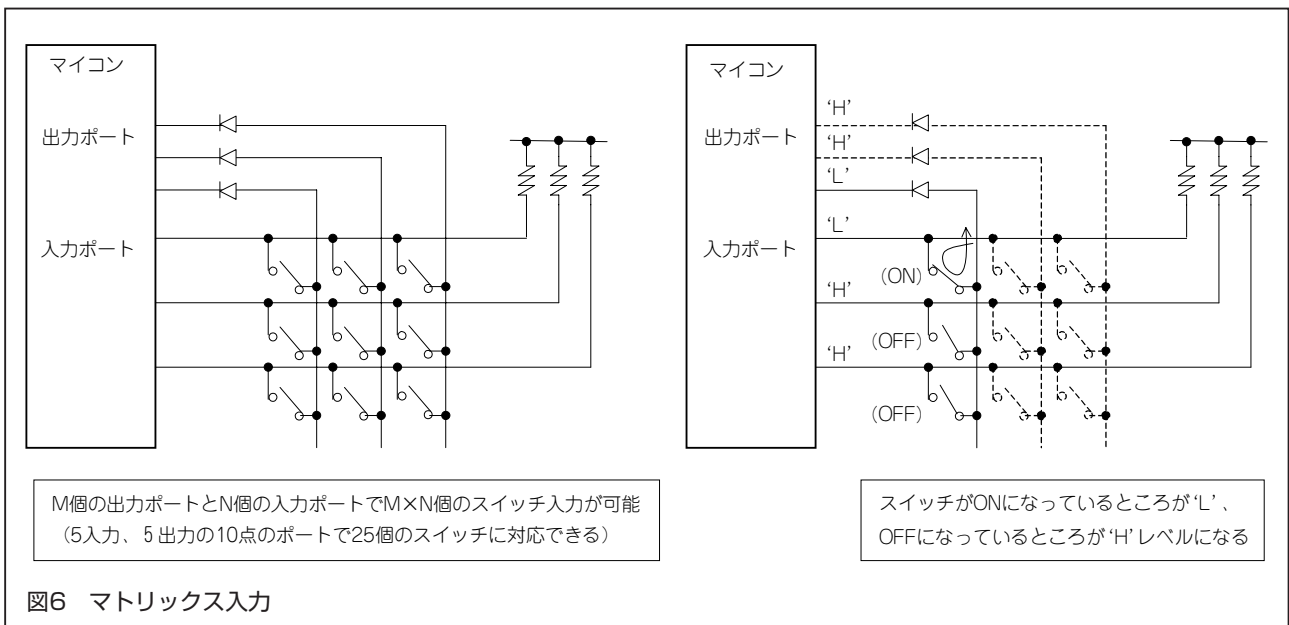
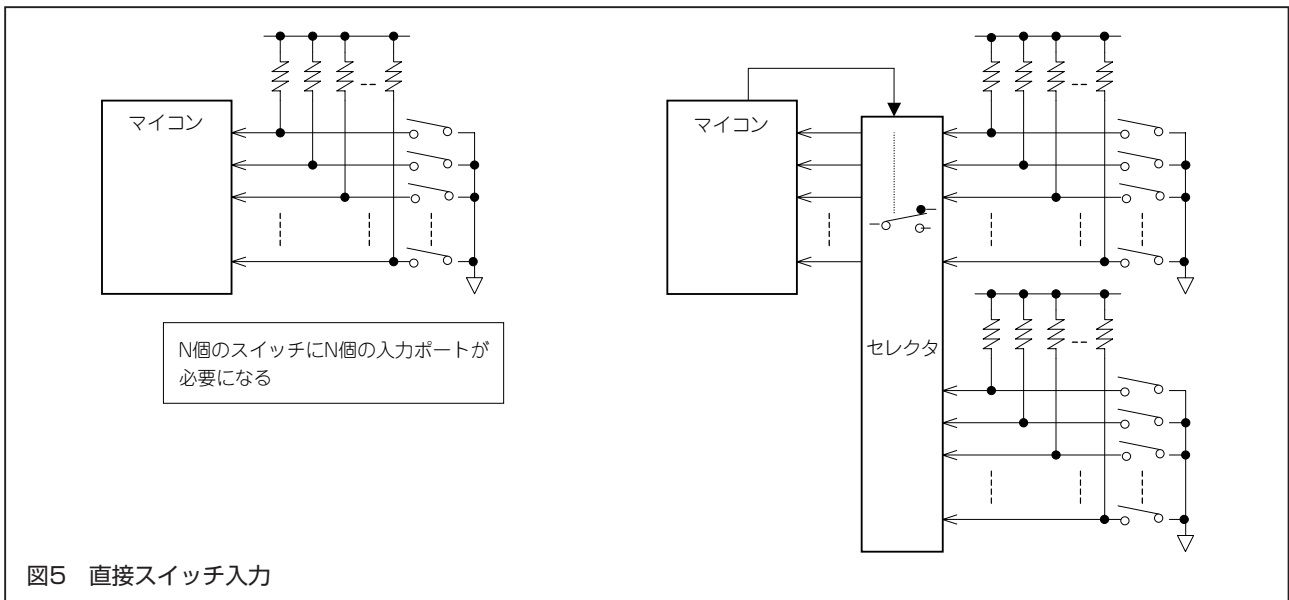
1) 直接入力

マイコンとスイッチの接続方法としてもっとも簡単なものは図5のように入力ポートに直接スイッチをつないでいく方法です。それぞれのスイッチの配線は独立していますので、配線の自由度が高く、読み取り方法も単にポートを読めばよいだけという簡便さは利点ですが、スイッチの数だけ入力ポートが必要になりますので、スイッチが多数必要な用途ではこれが欠点となります。多少足りない程度であれば外部にデータセクタを付けるなどして、ポート数を増やすという方法もありますが、ハードウェアの規模が大きくなってしまい、せっかくのワンチップマイコンの小型、低コストという特長が薄まってしまいます。

2) マトリックス入力

PCのキーボードや操作パネルのように、比較的狭いエリアに多数のスイッチが必要とされる場合によく利用されるのが、スイッチを網目状に配線する方法です。図6の上の図のように交差した配線の交点部分にスイッチを配置していくのです。先ほどの直接入力と異なるのはスイッチの片側がGNDに直結ではなく、マイコンの出力ポートとつながっているところです。

ここで図6の右側のように、出力ポートのひとつだけを「L」レベルにしてみます。この出力ポートがドライブしているポートの縦線上に並んだスイッチを押すと矢印方向に電流が流れ、入力ポートが「L」レベルになります。押され



ていないところは 'H' レベルのままです。仮にこの縦線上以外のスイッチ (図では点線で示しています) が押されたとしても、このとき、点線側はオープン状態ですので、入力ポートには影響を与えません。

つまり、あらかじめ縦線のスイッチ列のどれを読むか決めて、そのデータを入力ポートで読むことで、縦線数×横線数のスイッチを読み取ることができるわけです。基本的な考え方は、直接入力の際の外部セレクタ付きに近いものがあると言えるかもしれません。なお、この回路では3個以上のスイッチが同時に押されたときに誤認する可能性があります。これが問題となるときにはそれぞれのスイッチに直列にダイオードを入れます。

この方法ではたとえば入力3ビット、出力3ビットならば9個、4ビットずつなら16個という具合に2N個の端子でN²個のスイッチを扱えます。今回のように30個の入力が必要な場合には5×6という構成で11本のポートが必要です。

3) アナログ信号として読む

マイコンにスイッチをつなぐ方法としてはここまで説明した二つの方法が一般的であり、正統派といえるでしょう。しかし今回のように30個ものスイッチ入力を行うにはマトリックス入力を行ったとしても5×6という構成になり、11個のI/Oポートが必要です。他に液晶ディスプレイなども付けることを考えると、20ピンパッケージのμPD78F9222ではピン数が足りません。

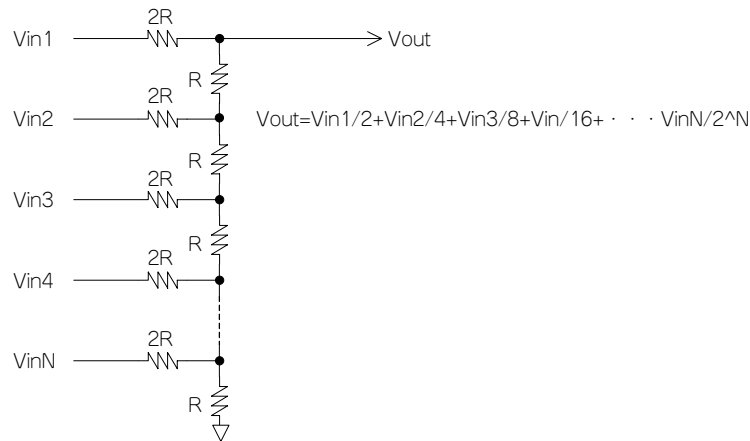


図7 ラダー型D/Aコンバータ

一般的な方法ではスイッチのON/OFF状態を‘H’レベルと‘L’レベル(‘1’と‘0’)の二値信号とし、これを0V、1V、2V・・・という具合にアナログ的に取り込むという考え方があります。μPD78F9222には分解能が10ビットのA/Dコンバータが内蔵されているので、ON/OFFというデジタル的なスイッチ情報をアナログ値に変換して、これをA/Dコンバータで読むという考え方です。

μPD78F9222内蔵のA/Dコンバータの分解能は10ビット(1024段階)ですので、フルに使えば1024個のスイッチの状態を読めるという理屈になります。もちろん、現実には変換エラーや誤差なども生じますのでここまできちんと行うのは無理ですが、少ないピン数でもかなり多くのスイッチをつなぐことができる可能性のある方法と言えるでしょう。

4) スイッチ状態をアナログ値へ

デジタル値からアナログ値への変換としてよく利用されるのが、図7のラダー型のD/Aコンバータです。抵抗値Rの抵抗と2倍の2Rの抵抗を図のように梯子型につなぎ、入力にVin1～VinNの電圧を与えると図の式のような電圧が得られます。入力をスイッチにして、Vcc、またはGNDとつながるようにすれば、 2^N の重み付けをしたデータになるわけです。

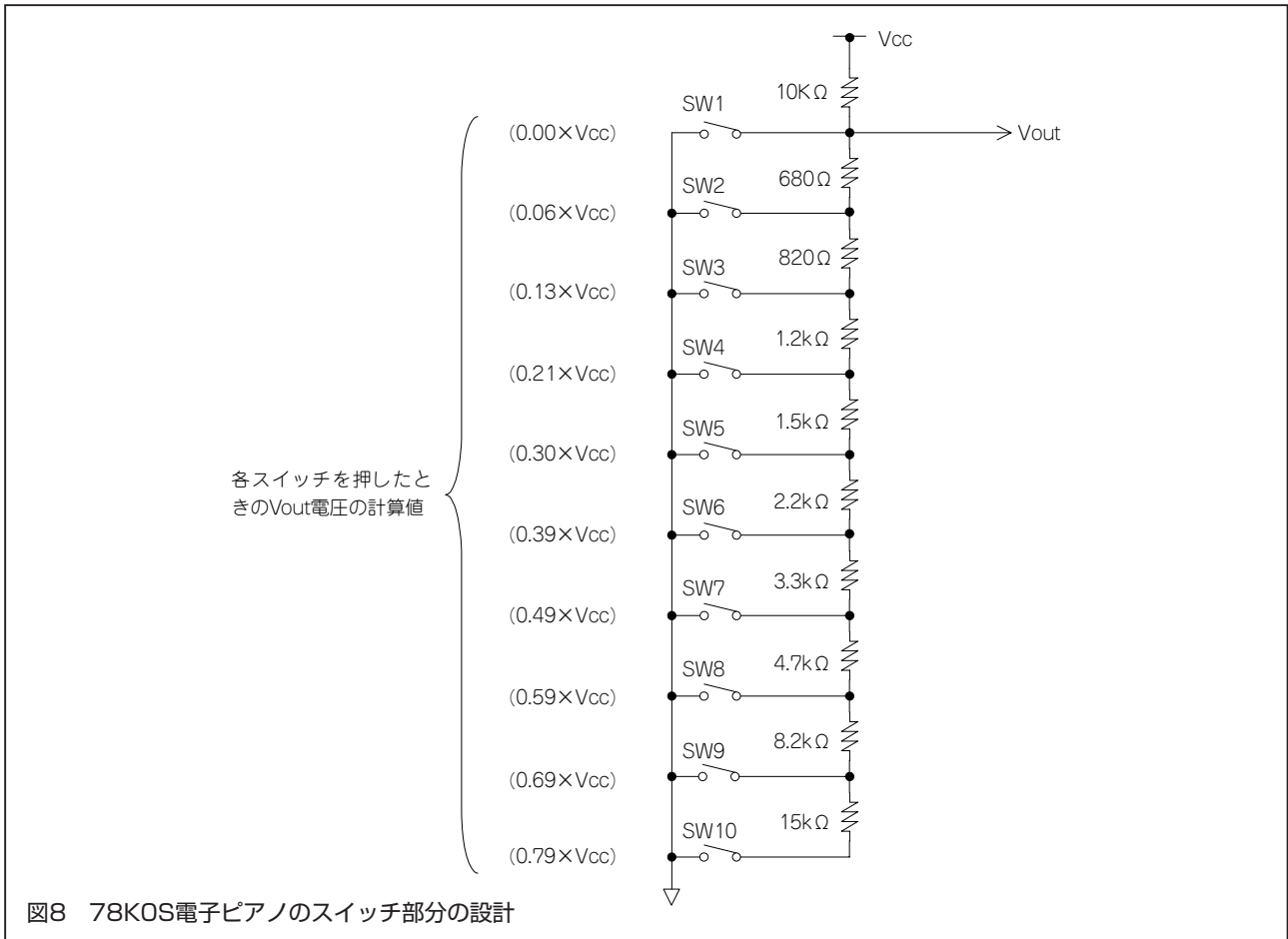
D/A変換ですので、当然のことながら複数のスイッチの同時押しにも対応できるという方法ですが、ビット数(スイッチ数)を増やすと抵抗の精度が問題になります。たとえば10個のスイッチでは10ビットのアナログ値となりますので、誤差0.1%も許されません。一般的に手に入れやすい抵抗はせいぜい5%程度ですので、まさに桁違いの精度が必要になります。

今回は音出力が1音だけですので、同時押しには対応しないという前提で、これを簡略化し、単に10個のスイッチで電圧を10等分するように設計しました。図8は今回設計したスイッチ入力部分の回路で、図の左側の値はそれぞれのスイッチを押したときのVout端子の電圧がどうなるかを算出したものです。最初が0.06、0.13と少し間が詰まっていますが、以降はほぼ $0.1 \times V_{cc}$ ずつ上昇します。全てOFF状態ならもちろんVccです。許容誤差は10%程度ありますので、制度5%程度の抵抗でも十分でしょう。

このVout出力電圧をμPD78F9222のA/Dコンバータで読み取れば、10個のスイッチが1つのアナログ入力端子で読めるという理屈です。今回はAIN0～AIN2の3つのアナログ入力端子にそれぞれ同じ回路を接続して合計30個のスイッチ入力に対応させました。わずか3ピンで30個のスイッチ入力ですから、他の用途でも応用がきくと思います。

● シリアルEEPROMのリード/ライト

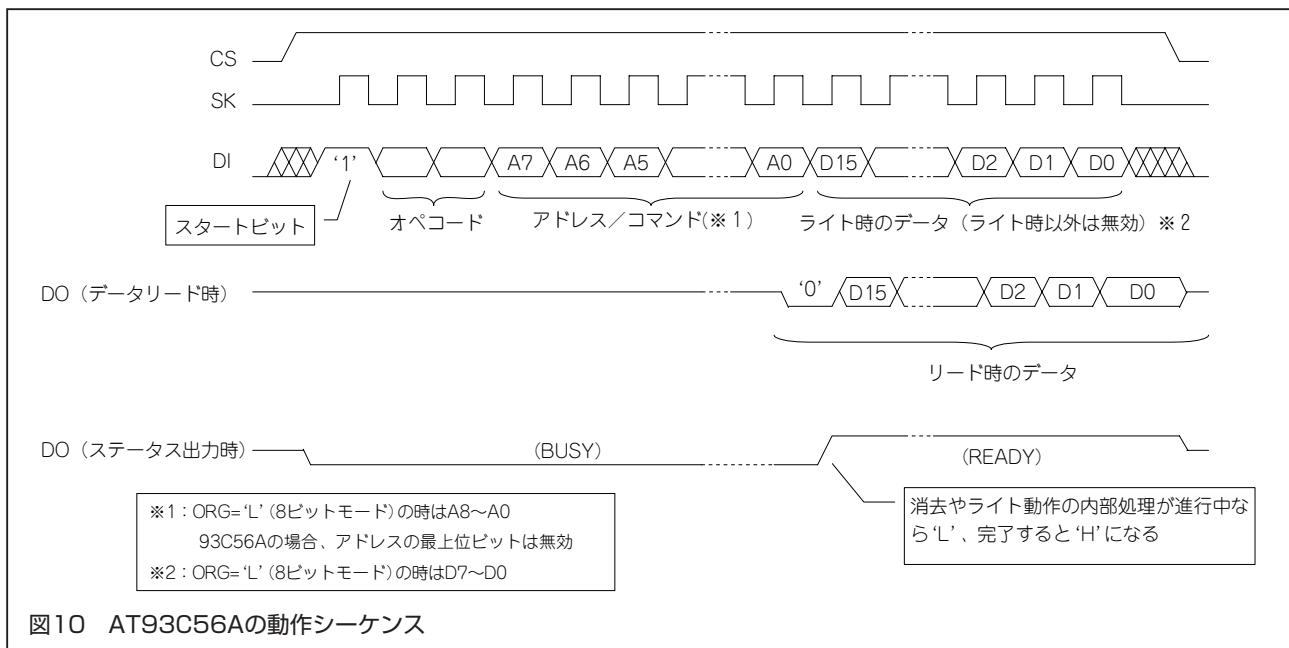
データを1ビットずつリード/ライトするようになってきている、シリアルEEPROMは8ピン程度の小さいパッケージのものが普通で、省スペースであることやピン数の少ないワンチップマイコンなどとも簡単に接続できること、不揮発性で電源が切れても記憶内容が保持されること、1バイト(ないし1ワード)単位での書き換えが行えること、などの特徴があります。これらの特徴を生かして外部データの取り込みを行う、いわゆるデータロガー的な用途のほか、装置の設定情報の記録などにも広く使われています。



シリアルEEPROMの外部インターフェースとしては、I²C(アイ・スケア・シーと発音します)やSPI、MicroWireの三種類がよく利用されています。これらのインターフェースはEEPROM専用というわけではなく、I/Oデバイスなど色々なものがつなげられるように配慮されていることもあり、やや扱いが面倒な面もあります。

今回使用したATMEL社のAT93C56Aはこれらに比べてシンプルな制御方法をとったEEPROMで、今回のように先頭アドレスから順次読み出していくような用途に向けたものになっています。

AT93C56Aとのインターフェース部分の回路は図9のようになっています。SK(シリアルクロック)がコマンドやデータのやり取り用のタイミング信号、CS(チップセレクト)がシリアルEEPROMの選択信号、DI/DOがそれぞれデータの入出力端子です。ORG(Organization)端子は、メモリアクセスを8ビット単位で行うか16ビット単位で行うかの切り替えです。AT93C56Aの容量は2Kビットですので、16ビット単位にすると、128×16ビットという構成になりアドレス指定が00h番地から7Fh番地まで、8ビット単位にすると256×8ビットという構成になり、アドレスは00h番地からFFh番地までになります。ORG端子を'H'レベルまたはオープンにしておくと16ビット、'L'レベルにすると8ビット単位になります。今回は1つのキー情報を16ビットで表現したので、16ビット単位でアクセスするこ



とにしました。ORGはオープンにしていますが、気になる方はプルアップしておけばよいでしょう。
CSはデバイスの選択信号で、'H'レベルのときにコマンド等に応答します。

1) データ入出力動作

図10はシリアルEEPROMのアクセス動作を図にしたものです。実はAT93C56Aと、容量が2倍のAT93C66Aで共通になっています。

マイコン側からのコマンドやデータの取り込みはクロック(SK端子)の立ち上がりエッジで行われます。また、EEPROMからのデータやステータスは、クロックの立ち上がりエッジから出力され始めます。

マイコン側からのコマンドは、まずスタートビット('1'='H')を与え、続いて2ビットのオペコードを送ります。この後はアドレスフィールドになり、ORG='H'(16ビットモード)の時には8ビット分のアドレス情報を与えます。メモリのリード/ライトコマンドの時にはこのフィールドはアクセスするアドレス指定(ただし、AT93C56Aの場合には最上位ビットは無効)になります。そのほかの消去コマンドなどではこのフィールドがコマンドの一部として扱われます。

データライト時はこのあとに書き込みたいデータを与えます。今回は16ビットモードにしていますので、16ビット分のデータを上位アドレスから順に与えます。

リード時にはアドレスの最下位ビットを与えたときにEEPROMのDO出力が'0' (= 'L')になり、次のクロックの立ち上がりエッジから順に上位ビットから出力されます。

消去(ERASE)や書込み(WRITE)コマンドの場合にはコマンド発行後、動作完了まで最大10ms程度の時間をとらなくてはなりませんので、これらのコマンドの後にはDO端子が内部処理中か否かのステータス出力として動作するようになっています。

図のように、内部処理中はDO端子が'L'レベルになり、内部処理が完了してレディ状態になると'H'レベルになります。

2) AT93C56Aのコマンド

AT93C56Aのコマンドとそのフォーマットは表1のようになっています。ERASEコマンドは1アドレス単位での消去、ERALコマンドはチップ全体の消去、WRALはチップ全体に固定データを書き込むコマンドです。

注意が必要な書込みは

- 1) ライトイネーブルコマンド発行
- 2) 書込みコマンド発行
- 3) ライトディセーブルコマンド発行

表1 シリアルEEPROM (AT93C56A) のコマンド

コマンド名	意味	スタートビット	オペコード	アドレス	データ		備考
					データIN	データOUT	
		1ビット	2ビット	8ビット(※1)	16ビット(※2)	16ビット(※2)	
READ	リード	'1'	'10'	A7~A0	—	D15~D0	指定したアドレスのデータを読み出す
EWEN	ライトイネーブル	'1'	'00'	'11XXXXXX'	—	—	書き込みコマンドイネーブル(データを書き替えるコマンドの前に必須)
ERASE	1ワード消去	'1'	'11'	A7~A0	—	—	データ書き込み(DO='0':内部書き込み動作中/'1':内部書き込み動作完了)
WRITE	書き込み	'1'	'01'	A7~A0	D15~D0	(内部動作ステータス)	指定したアドレスのデータ消去(FFhにする)
ERAL	イレース・オール	'1'	'00'	'10XXXXXX'	—	(内部動作ステータス)	チップ全体を一括消去(FFhにする)
WRAL	ライト・オール	'1'	'00'	'01XXXXXX'	D15~D0	(内部動作ステータス)	チップ全体に指定データを書き込む
EWDS	ライトディセーブル	'1'	'00'	'00XXXXXX'	—	—	書き込みコマンドディセーブル(データ書き替えの後に実行する)

※1: ×16ビットモード時。×8ビットモード時は9ビット
 ※2: ×16ビットモード時。×8ビットモード時は8ビット

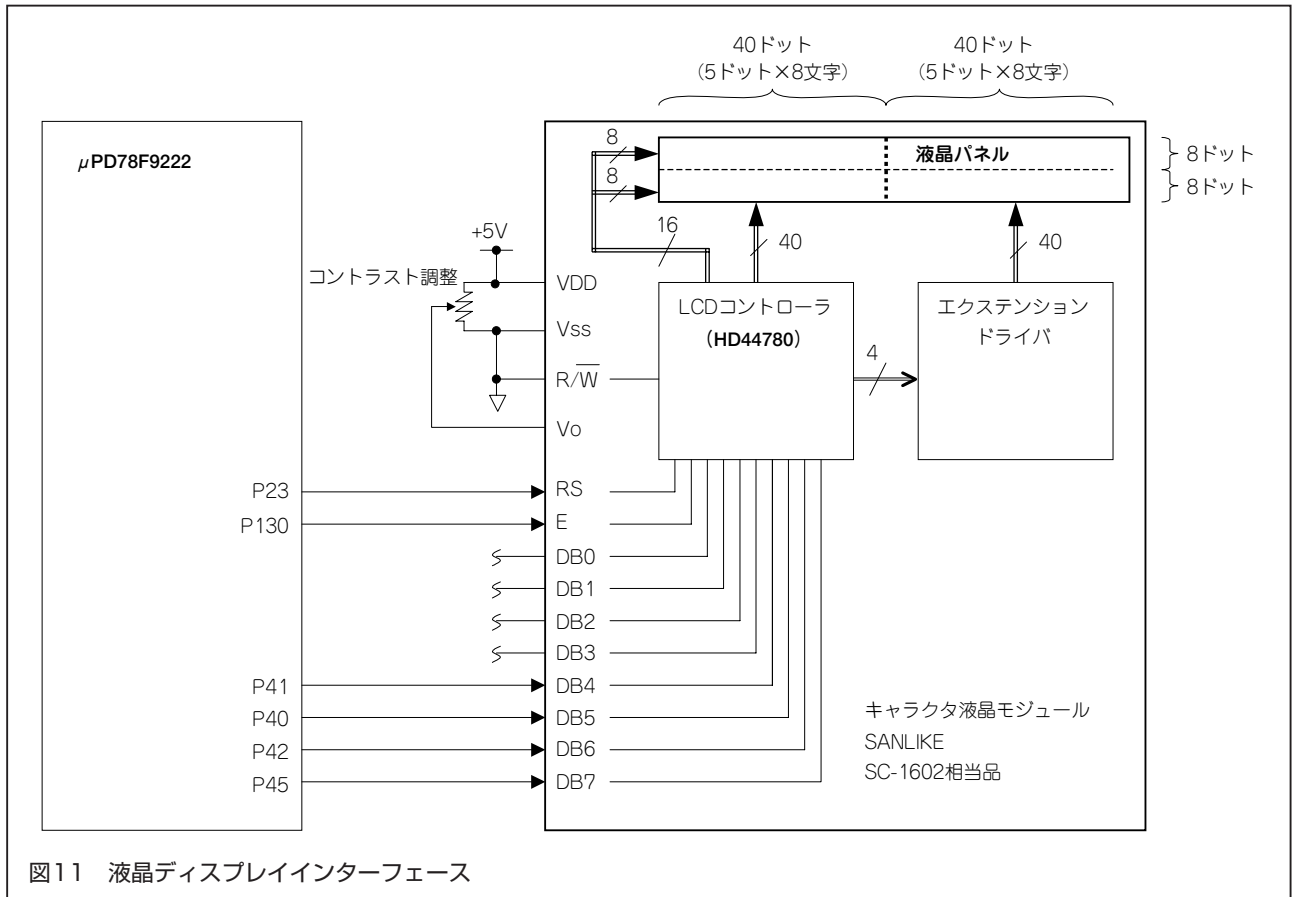


図11 液晶ディスプレイインターフェース

という3ステップで行われるということです。これは不用意な書き換えから保護するための仕掛けです。リード方向はこのような手間はありませので、READコマンドだけで読み出しが可能です。

●キャラクタ液晶ディスプレイの制御

図11はキャラクタ液晶パネル部分の回路です。キャラクタ液晶ディスプレイはマトリクス状になった液晶パネルと、キャラクタ表示用の液晶コントローラが一体になったものです。方式などはいろいろと考えられますが、日立の液晶コントローラHD44780に準拠したものが事実上の標準(デファクトスタンダード)として広く使われています。今回使用しているSANLIKE社のSC-1602も同様にHD44780に準拠したインターフェースになっています。

HD44780は液晶パネル制御回路のほか、80バイトのディスプレイメモリや文字フォント用のROM、ユーザ用のフォントRAMを持っており、ディスプレイメモリに文字コードを書き込めば、対応するフォントデータが自動的に読み込まれて表示されるため、小規模なワンチップマイコンでもそれほど大きな負荷にならない表示装置として便利です。

ディスプレイメモリは80文字分ありますが、単体での液晶ドライブ能力は横40ドット、縦16ドットですので、5×

8ドット(横×縦)のフォントを使ったとしても8文字×2行の計16文字分ですので、HD44780は外部にエクステンション・ドライバと呼ばれる横方向のドット数拡張のためのデータラッチ回路を付けることができるようになっています。これを利用することで横80ドット(16文字)まで拡張され、計32文字まで一度に表示できるようになります。今回使用したSC-1602もエクステンション・ドライバが入っていて、横16文字、縦2行の表示が行えるようになっています。

●R/W(リード/ライト)

データ転送方向を指示するもので、コマンドやデータの書き込み方向ならば‘L’、データやステータスのリードならば‘H’にします。今回はライト方向しか使わないので、‘L’レベルに固定して、常にライト状態にしています。マイコン/I/Oピンも節約できました。

●RS(レジスタセレクト)

コマンド(HD44780内部にある制御用のレジスタをアクセスする)なのか、ディスプレイ用やキャラクタフォント用のRAMをアクセスするのかを指定するものです。‘L’だとコマンド、‘H’ならRAMアクセスになります。

●E(Enable Clock)

リード/ライト動作(今回はライト方向しかありませんが)の開始を指示するものです。コマンドやデータラッチ用のクロック信号と思えば良いでしょう。E信号の立下りエッジで、データやコマンドが受け取られます。

●DB(データバス)

データやコマンドなどのやり取りを行うためのデータバスです。HD44780の内部レジスタやRAMなど8ビット構成になっていますが、特に小規模なマイコンではデータのために8本のI/Oピンを取られるのが厳しいことから、4ビットずつアクセスする4ビットモードも備えています。今回も4ビットモードで使用しました。4ビットモードの時にはDB4～DB7が使われます。

●キャラクタ液晶ディスプレイのアクセス方法

図12は液晶ディスプレイに表示データ(文字コード)を書き込む時の動作を示したものです。まず4ビット転送2回でディスプレイデータRAM(DDRAM)のアドレス設定を行います。コマンドなのかデータライトなのかは先ほど説明したとおり、RS信号で区別されます。この例ではRSが‘L’になっていますのでコマンドになります。コマンド表を見るとわかりますが、8ビットのコマンドのビット7が‘1’ならばSet DDRAM addressコマンドで、下位7ビットがアドレスになります。

続いて文字コードの書き込みを行います。データ書き込みを行うと、今設定したアドレスから順に、RS信号を‘H’にして、データの書き込みを行います。今回は16ビットモードですので、データを4ビットずつ4回書き込みます。

8ビットモードにしていたときは、図12のように書き込むと、指定したアドレスのところに最初の8ビット分が書き込まれ、次の8ビット分は+1または-1されたアドレス(どちらになるかは、Entry mode setコマンドで行います)に書き込まれます。つまり、連続したアドレスに次々に書き込む場合には先頭アドレスを指定すれば、あとはデータだけを書き込み続けることができるのです。

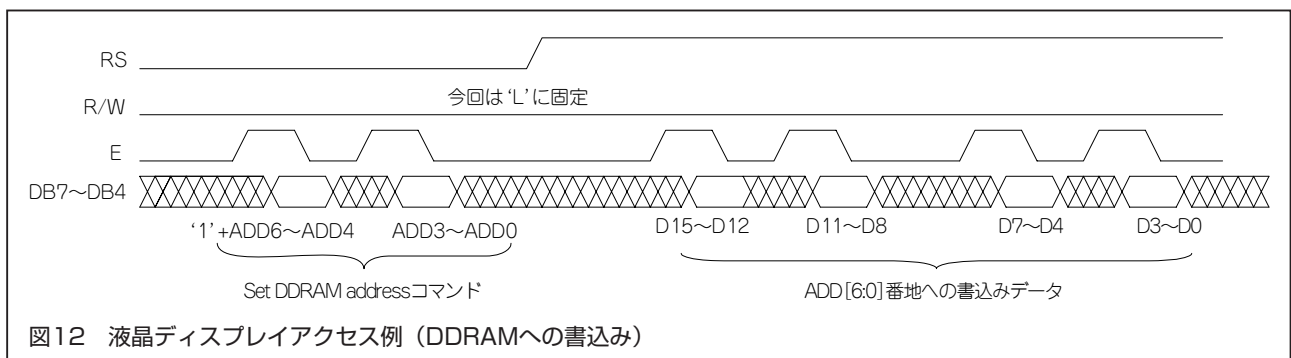


図12 液晶ディスプレイアクセス例 (DDRAMへの書き込み)

表2 液晶ディスプレイのコマンド一覧

コマンド名	R/S	R/W	DATA				機能	実行時間
			bit7	bit6	bit5	bit4		
Clear display	0	0	0	0	0	0	ディスプレイをクリアし、DDRAMアドレスカウンタを0にする	
	0	0	0	0	0	1		
Return home	0	0	0	0	0	0	DDRAMアドレスカウンタを0にし、表示位置も初期化 ただし、ディスプレイRAMのクリアは行わない	1.52ms
	0	0	0	0	1	X		
Entry mode set	0	0	0	0	0	0	I/D: リードライト時のアドレス自動更新方向 ('1': インクリメント '0': デクリメント) S: リード/ライト時の表示シフト '1': シフトする	37μs
	0	0	0	1	I/D	S		
Display on/off control	0	0	0	0	0	0	D: 表示 ON/OFF ('1': ON) C: カーソル表示 ('1': ON) B: ブリンク ('1': カーソル位置のキャラクタが点滅する)	37μs
	0	0	1	D	C	B		
Cursor or display shift	0	0	0	0	0	1	S/C: 表示シフト/カーソル移動 ('1': 表示シフト '0': カーソル移動) R/L: 移動方向 ('1': 右, '0': 左)	37μs
	0	0	S/C	R/L	X	X		
Function set	0	0	0	0	1	DL	DL: データバス幅 ('1': 8bit '0': 4bit) N: 表示ライン数 ('1': 2ライン '0': 1ライン) F: 文字フォント選択 ('1': 5×10ドット, '0': 5×8ドット)	37μs
	0	0	N	F	X	X		
Set CGRAM address	0	0	0	1	ACG5	ACG4	CGRAM(フォントパターンRAM)のアドレスセット。 以降のデータリード/ライトはCGRAMに対して行われる	37μs
	0	0	ACG3	ACG2	ACG1	ACG0		
Set DDRAM address	0	0	1	ADD6	ADD5	ADD4	DDRAM (表示データRAM)のアドレスセット。 以降のデータリード/ライトはDDRAMに対して行われる	37μs
	0	0	ADD3	ADD2	ADD1	ADD0		
Read busy flag & address	0	1	BF	AC6	AC5	AC4	BF: 内部動作中 (ビジー) フラグ ('1': ビジー '0': レディ) AC6 ~ AC0: アクセス中のアドレス (CGRAM/DDRAM 共用) 0	0μs
	0	1	AC3	AC2	AC1	AC0		
Write data to CG or DDRAM	1	0	D7	D6	D5	D4	CGRAM/DDRAM へのデータ書込み (いずれになるかは SetCGRAM address/SetDDRAM address によって決まる)	37μs
	1	0	D3	D2	D1	D0		
Read data from CG or DDRAM	1	1	D7	D6	D5	D4	CGRAM/DDRAM からのデータ読み込み (いずれになるかは SetCGRAM address/SetDDRAM address によって決まる)	37μs
	1	1	D3	D2	D1	D0		

●液晶ディスプレイのコマンド

液晶ディスプレイの制御コマンドは表2のようになっています。数が多くて面倒そうに見えますが、ほとんどは電源投入後の初期設定に使われるだけです。初期設定はそれほど変更する必要は無いので、そのまま流用して、先ほどのタイミング図で示したSet DDRAM addressとWrite data to CG or DDRAMだけ理解しておけばよいでしょう。

コマンド表の右にある時間はコマンドの処理にかかる時間です。コマンド発行後ここに書かれた時間だけ待ってから次のコマンドを発行しなければなりません。

●初期化手順

液晶ディスプレイの初期化手順を図にしたのが図13です。入り組んで見えるのは、データバス幅を4ビット幅に設定するまでの手間がかかっているためです。なお、この設定までの手順はデータシートを基にしたもので、実際のプログラムとは少々異なっている部分もありますので、注意してください。

今回のピアノソフトウェアでの初期設定データは図3のフローチャートの中に書いてあります。先ほどのコマンド表とつき合わせてみれば、わかりやすいと思います。

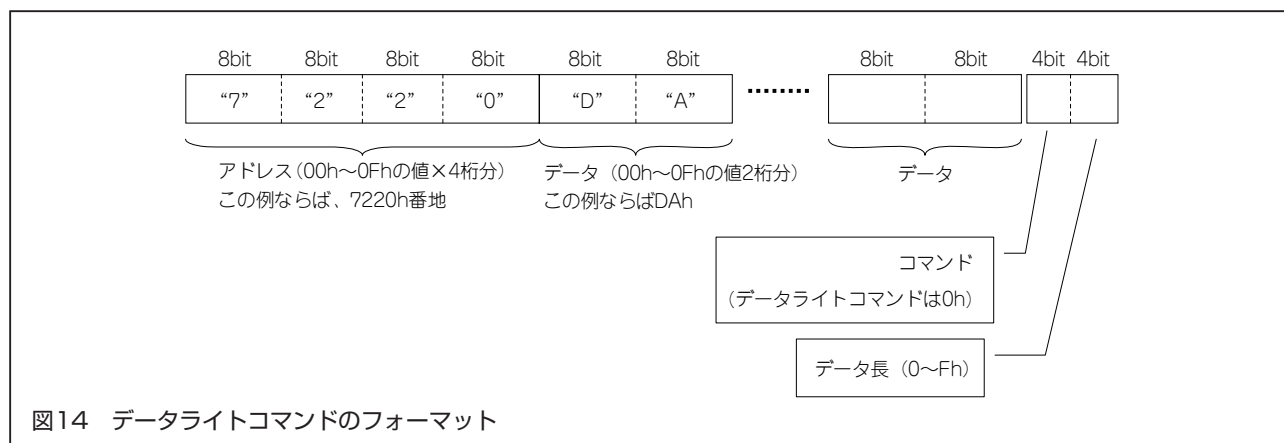
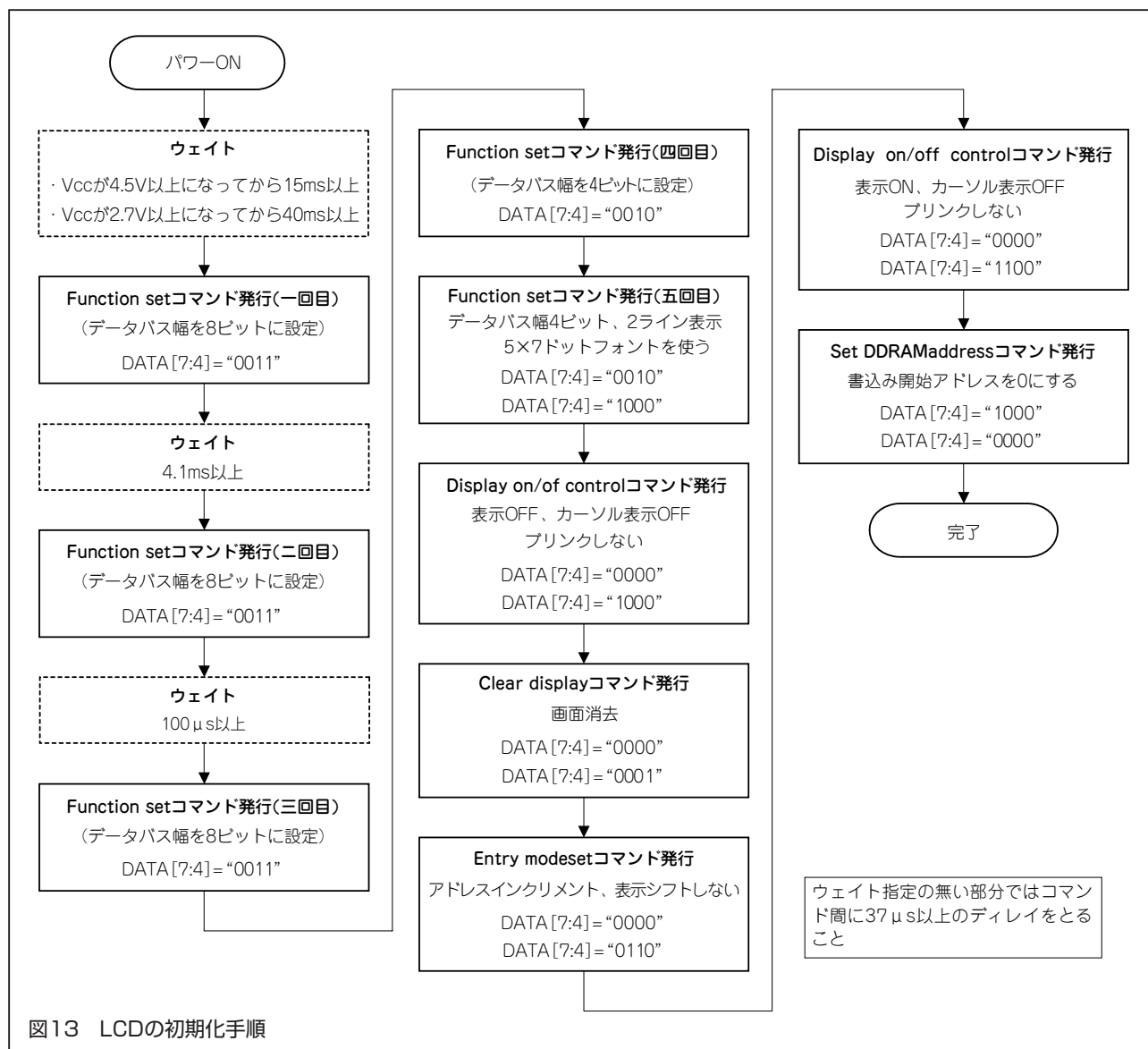
●メモリファインダー

機能的にはこれで十分ですが、今回はデバッグやテストに便利なメモリファインダー機能を追加しました。

メモリファインダーは、動作中にシリアルポート経由でメモリのリード/ライトやビットセット/リセットを行えるもので、今回は4つの機能を持っています。

- 1) メモリの書込み(一回に最大16バイト)
- 2) メモリの読み込み(一回に最大16バイト)
- 3) メモリのビットクリア
- 4) メモリのビットセット

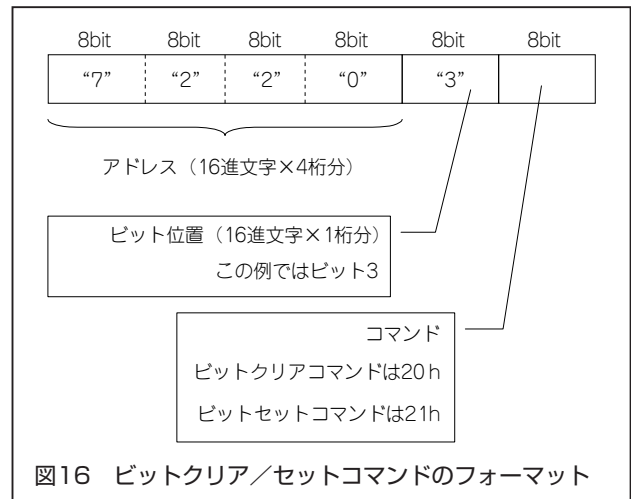
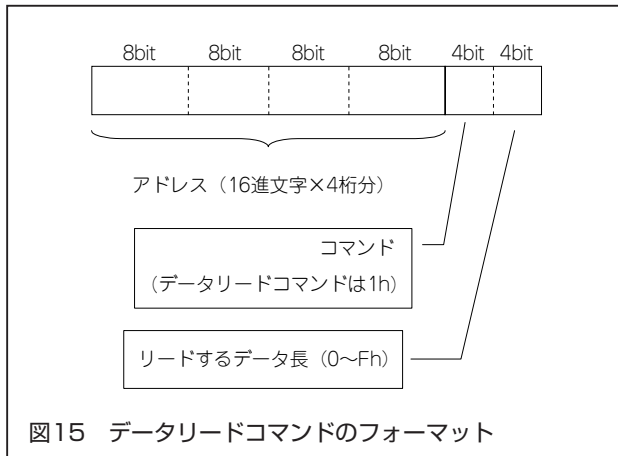
メモリファインダー機能はピアノ機能を動作させた状態のまま使うことができ、メモリやSFRの設定状態やカウンタの動作状態を見ることができ、独自の機能追加や変更などを行なったときの動作確認や、思ったように動かない場合のチェック、動作テスト用等に便利に利用できます。



●メモリライトコマンド

メモリライトコマンドのフォーマットは図14のようになっています。書き込みを開始する先頭アドレスとデータはASCII文字列で与え、最後の1バイトでコマンドと、データサイズを示します。

上位4ビットが0だとデータライトコマンドという扱いになり、下位4ビットでデータ数(バイト数)を示します。図は7220番地にDAhを・・・という具合に書き込む要求の例です。なお、実際にあたえたデータ数と、最終バイトで指定したデータ数が一致しない場合にはエラーになります。



●メモリリードコマンド

メモリリードコマンドのフォーマットは図15のようになっています。ライトコマンドと同様に最初の4バイトはASCII文字で、アドレスを与え、5バイト目でコマンド、データ数を指定します。メモリリードは上位4ビットが1です。読み出されたデータはASCII文字列にしてシリアルポート経由でPC側に渡します。

●ビットクリア/セットコマンド

ビットクリア/セットコマンドのフォーマットは図16のようになっており、指定したアドレスの8ビットデータのうち特定のビットを'0'(クリア)、または'1'(セット)にします。

最初の4文字分がアドレス、次の1文字がクリアやセットを行いたいビット位置、最後がコマンドで、20hならばビットクリアコマンド、21hならばビットセットコマンドになります。

製作と実行

製作上では特に難しいところはないと思いますが、スイッチの配線が少々多くて大変かもしれません。アナログ信号を扱う部分については配線が極端に長くならないようにしたほうが良いと思います。

出来上がったら、配線チェックをしてファームウェアを書き込んで動かしてみましょう。電源投入後のデフォルトでは演奏(PLAY)モードになっていますので、スイッチを押すと対応する音階の音が出るはずですよ。

ここで、MODEスイッチを押して録音(RECORD)モードに切り替えます。スタートスイッチを押してから演奏して、ストップボタンを押せばEEPROMに演奏データ格納が終了します。

再びMODEスイッチを押して再生(REPLAY)モードに切り替えてスタートスイッチを押すと今演奏したものが再生されるでしょう。

おわりに

「単音出力の電子ピアノ」とだけ聞くと、機能的に物足りないと思われた方も少なくないと思います。しかし、中を見てみれば20ピンという小さなマイコンでありながら外部にIC類を追加することなく30個のスイッチ入力を可能にしたことや、液晶ディスプレイ表示、シリアルEEPROMへの記録/再生、動作させたままのデバッグに便利なメモリファインダーなど、広く応用可能な要素を数多く含んだ、面白い製作事例ではないかと思えます。

編集 桑野 雅彦