応用編

第6章

アナログ信号の出力

…D-Aコンバータを接続してアナログ電圧や正弦波を出力する

A-Dコンバータによるアナログ・データ入力ができるようになったら，やはり次にやりたくなるのは逆向き，つまりD-A変換でしょう．最も単純なD-Aといえば，単なるICの出力ピンです．UCT-203ボードでもPIOモードで‘0’をセットすればLレベル，‘1’にすればHレベルの信号が出るわけですから，1ビットのD-Aコンバータと言えなくもありません．また，第4章で行ったPWM制御なども最終的なモータの回転数というアナログ的な値を変化させています．実際，このパルスを積分して値を取ればアナログ電圧として取り出すことはできます．

ただ，言うまでもなく，これではまったく面白くありません．やはりD-A変換と言うからには，値をセットしたらすぐに0.1 Vとか，0.45 Vといったアナログ電圧を出力できるようでなくては面白みがないというものでしょう．

6-1　使用するD-Aコンバータ IC

一般的なD-Aコンバータは，入力として複数ビットのパラレル・データからアナログ的な電圧を作り出すというものです．この方法でよく使われるのが「*R*-2*R*ラダー」と呼ばれるものです．**図1**に回路を示します．抵抗が「はしご（ラダー）」のようにつながっていくのでこの名称が付いたのでしょう．*R*と書かれた抵抗はすべて同じ値，2*R*と書かれたものは*R*の2倍の抵抗値のものです．2*R*の抵抗は*R*の抵抗を2本直列にすればよいので，この回路は抵抗値*R*の抵抗一種類だけで構成することができます．

値の比率がそろった抵抗を用意するのに比べて，値が同じ抵抗を揃えるのは比較的簡単です．この回路では抵抗値自体はばらついていても，使用する抵抗の値が同じであればよいというのが利点です．たとえば集合抵抗などを使うと，値そのものの誤差はそれなりですが，1パッケージの中での各素子の値は比較的そろっているようです．

スイッチが値入力用で，一番上が上位ビットになります．現在はスイッチになっていますが，ここをCMOSのバッファICなどに置き換えればそのまま応用可能です．ただ，この場合には*R*の値をあまり小さくすると出力電圧が低下して，正しい値が出なくなりますので，注意が必要です．

また，この図では出力側はそのまま出していますが，出力側のインピーダンスが小さいと出力のところについている2*R*の値が変わったのと同じことになってしまいますので，やはり出力電圧が予定どおり出なくなってしまいます．OPアンプを使った非反転増幅回路を使うなどして受けるのがよいでしょう．

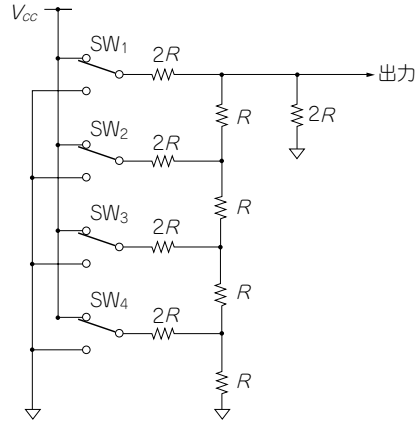

図1　*R-2R*ラダー型D-Aコンバータ

●D-Aコンバータ IC を使う

R-2R のラダー型D-Aコンバータは比較的簡単に作れますので，実験してみるのもよいでしょうが，ここではもう少し一般的にということで，専用のD-Aコンバータ IC を使ってみることにしました．

使用したのはナショナル・セミコンダクター社のDAC0800です．これも前章のADC804と同様，定番とも言えるD-Aコンバータ IC で，入手も比較的容易でしょう．ディジタル入力が与えられてからアナログ・データが制定するまでの時間は100 nsと，比較的高速です．

中の構造は**図2**のようになっています．D-A変換の鍵になっている部分は，やはり抵抗をラダー状に組んだものです．

●DAC0800について

DAC0800の中身が，単純な*R-2R*のラダーにOPアンプが付いたようなものであれば簡単なのですが，そのつもりでデータシートを読んでいくとよくわからないことになってくるのではないかと思います．実は，DAC0800は電流動作を基本にしているのです．前に示したラダーの場合には電圧動作が基本でしたが，DAC0800ではラダーにつながっているトランジスタの電流総和が一番左側のOPアンプにつながっているトランジスタに流れる電流と等しくなる（実際には255/256になる）ように動作するのです．

つまり，$V_{REF(+)}$に流れ込む電流が基準電流となって，これがDAC0800の二つの出力端子に流れる電流の総和に等しくなるわけです．二つの出力端子の和は常にV_{REF}に流れ込む電流と等しいので，一方が他方の反転出力ということになります．

図3に基本的な使いかたを示します．基準電流I_{ref}は，

$$I_{ref} = \frac{V_{ref}}{R_{ref}}$$

で計算されます（ズレはR_{15}で調整，それほど精度がいらないなら$R_{15} = R_{ref}$でよい）．出力電流の合計値（最大値

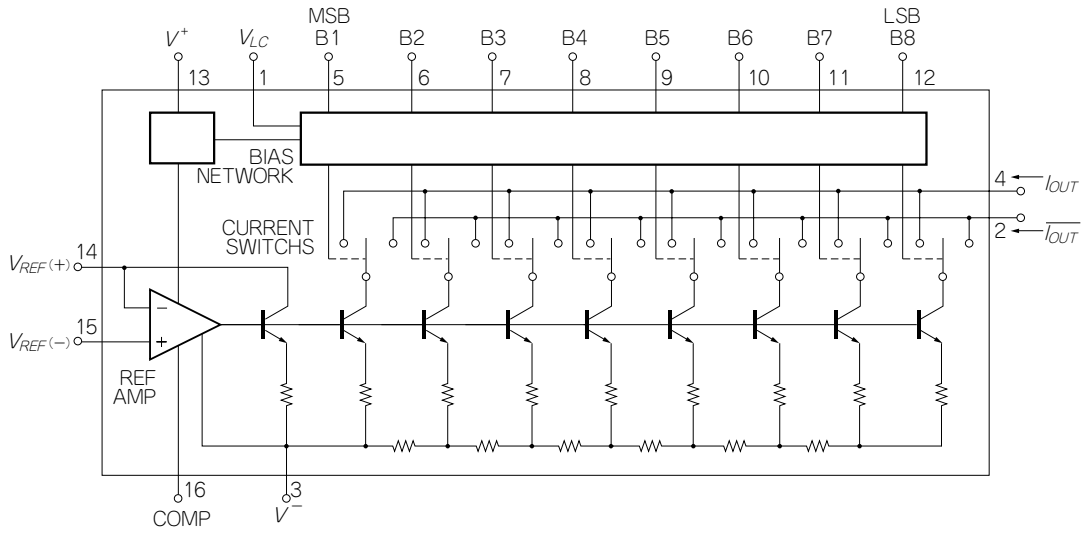

図2　DAC0800の内部ブロック

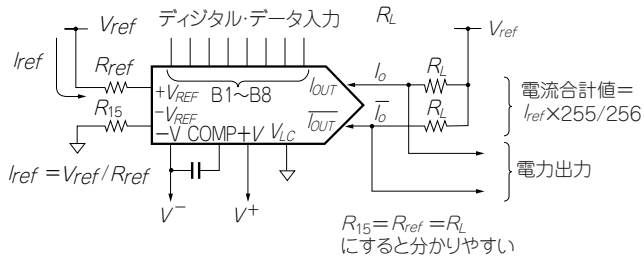

図3　DAC0800の基本的な使いかた

も）はI_{ref}×255/256になります．ここに，R_{ref}とR_{15}とR_Lを同じ値の抵抗にしておけば，出力は0V～V_{ref}までの範囲になりますので，V_{ref}を＋5Vにしておけば，＋5V系のアナログ出力が得られます．

V_{LC}ピンは単純に0Vにしておけば通常のTTLレベルを入力すればよいので，0V（GND）に接続しています．

6-2　拡張ハードウェアとプログラム

●回路作成

D-Aコンバータ基板の回路を**図4**に，試作した基板の外覧を**写真1**に示します．DAC0800を片電源動作させるのは難しそうなので，006Pの乾電池を2個使って±9Vの電源を作り，リファレンス電圧は3端子レギュレータで作成した＋5Vを利用しました．

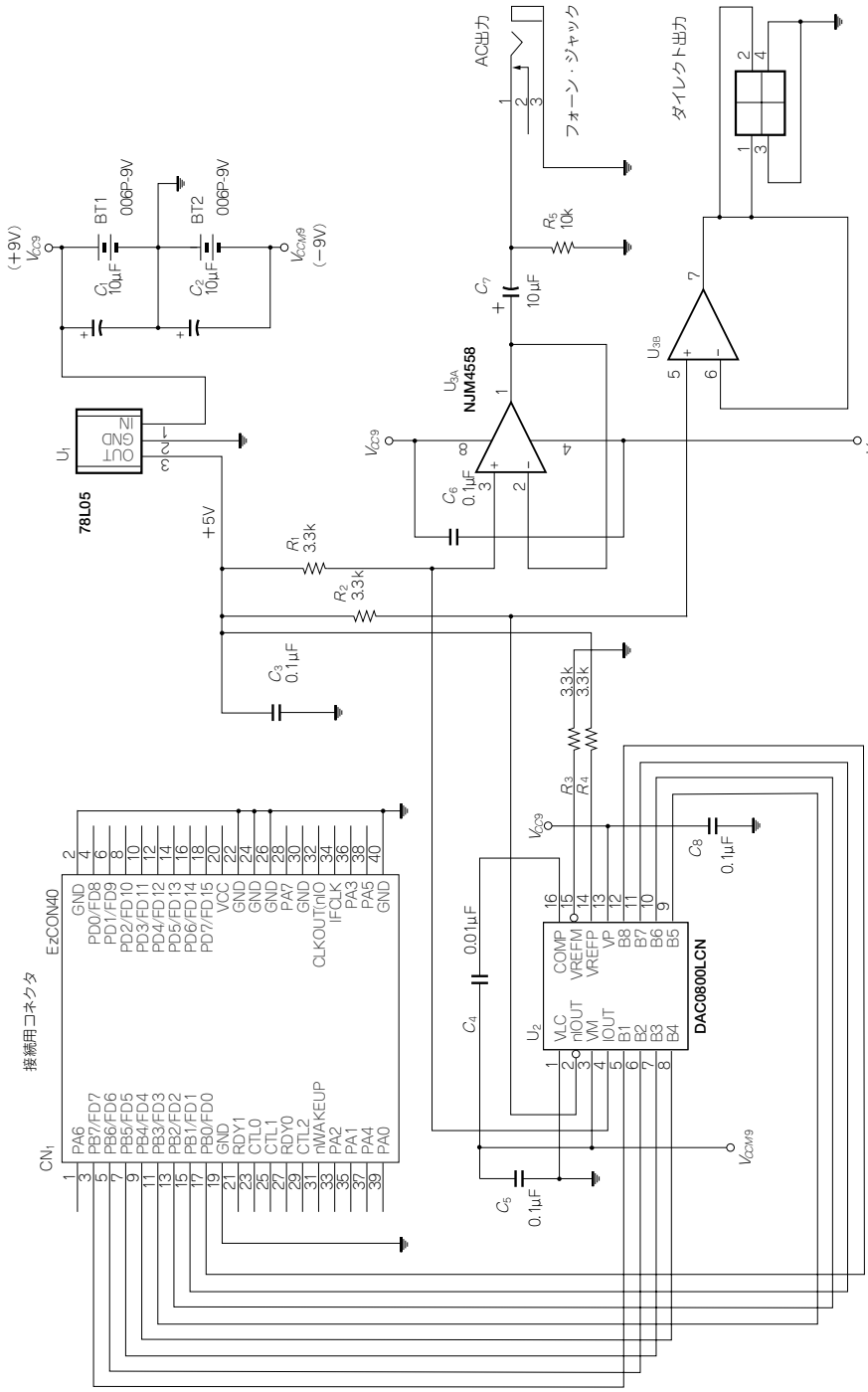

図4　D-Aコンバータ基板の回路

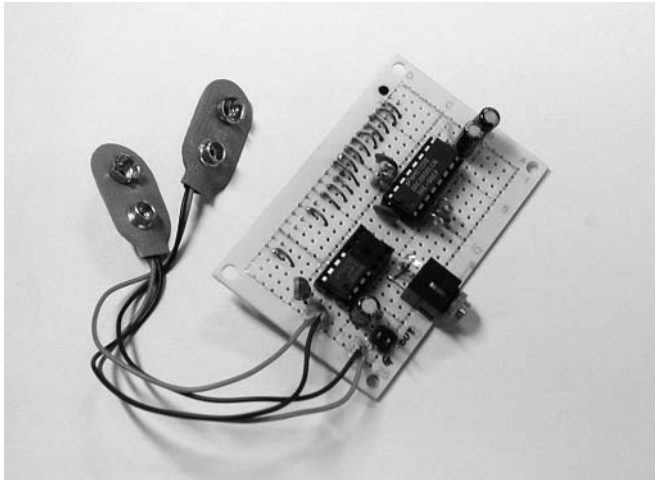

写真1　試作したD-Aコンバータ基板

図5　PIOモードでのD-Aコンバータ制御の実行例

データ出力はPORTBを使用しました．GPIFモードのときはデータ・バスの下位8ビット（FD［0：7］）になりますので，GPIFを使った連続データ転送にも使えるようにと考えたためです．

DAC0800の出力が二つあるので，出力を2回路入りのOPアンプIC（NJM4558，新日本無線）を使って，ダイレクト出力とコンデンサでDCぶんをカットしたAC出力の二つを用意してみました．AC出力のほうはオーディオ出力，ダイレクト出力は＋5V片電源動作させるタイプのターゲットを繋ぐことを想定しました．

ダイレクト出力のほうはゼロのときに0V，FFhのときにほぼ＋5Vになる反転出力側（$\overline{I_{OUT}}$，2番ピン）を使い，AC出力側は通常極性をあまり意識することはないので正出力側（I_{OUT}，4番ピン）を使っています．

■PIOモードのプログラム

まず，最初に作成したアプリケーションは**図5**のようなものです．スクロール・バーで出力電圧を変える機能と，［Auto］ボタンで自動的に20msのインターバルで00hからFFhまで掃引するモードをもたせてみました．

プログラム（EzDAC.FRM）の主要部は**リスト1**のようになります（付属CD-ROMのEzDACフォルダに収録）．**Form_Load()**の中で，**EZ_SetPortConfig()**でPIOモードに設定し，**EZ_PIOWrite()**でデータ出力を行います．

タイマ・コントロールのインターバルの初期値はゼロにしてありますので，最初はタイマ・イベントは発生しません．［Auto］ボタンが押されると，インターバルに20（20msの意味）をセットし，20msごとに**Timer1_Timer**の中で値をインクリメントしながら出力します．

00hからFFhまでのスイープで5秒程度かかりますので，アナログ・テスタなどで見ていても電圧が上下するのがよくわかるでしょう．

リスト1　PIOモードでのD-Aコンバータ制御の主要部（EzDAC.FRM）

```
'================================================================================
' D/Aコンバータ接続サンプル
'
' タイマー割込みでの自動インクリメントとスクロールバーによるマニュアル動作
'  PB[0..7] => DAC0800-B[8..1]
'
'  DAC0800の場合、B1がMSB、B8がLSBになっているので
'  配線をひっくり返している。
'================================================================================'
Dim DAData As Byte
Dim autorun As Byte
Dim Fullscale As Single
Sub Disp_Voltage(DAData As Byte)
    Text_DAVal.Text = Format((DAData + 1) * Fullscale / 256, "0.00")
End Sub

Private Sub Cmd_Autorun_Click()
    If (autorun = 0) Then
        autorun = 1
        Timer1.Interval = 20
    Else
        autorun = 0
        Timer1.Interval = 0
    End If
End Sub

Private Sub Form_Load()
    Dim sts As Long
    Fullscale = 5.14     ' フルスケール電圧（実測で決定）手元では5.14Vだった
    DAData = 0
    autorun = 0
    Timer1.Interval = 0
    Scrl_DAVal.Min = 0
    Scrl_DAVal.Max = 255
    Scrl_DAVal.value = DAData
    Disp_Voltage (DAData)
    Call EZ_Open
    sts = EZ_SetPortConfig(0, 2, 1, 1, 1, 1, 1)
    sts = EZ_PIOWrite(1, DAData)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Call EZ_Close
End Sub

Private Sub Scrl_DAVal_Change()
    DAData = Scrl_DAVal.value
    sts = EZ_PIOWrite(1, DAData)
    Disp_Voltage (DAData)
End Sub

Private Sub Timer1_Timer()
    sts = EZ_PIOWrite(1, DAData)
    Scrl_DAVal.value = DAData
    Disp_Voltage (DAData)
    If DAData = 255 Then
        DAData = 0
    Else
        DAData = DAData + 1
    End If
End Sub
```

リスト2　GPIFモードでのD-Aコンバータ制御の主要部（EzDACGPIF.FRM）

```
'================================================================================
' D/AコンバータＧＰＩＦ出力サンプル
'
'  ＧＰＩＦを使って波形出力する
'
'  FD[0..7]・・DAC0800のB[8..1]
'  FD[8..15]は未使用
'================================================================================
Dim DAdata(16383) As Byte
Dim wsw(31) As Byte
Dim xfrlen As Long
Dim start As Byte

Private Sub Cmd_Start_Click()
    Dim sts As Long
    Dim i As Integer
    sts = EZ_GPIFTrig(0, &H7FFFFFFF)
    Label_Cond.Caption = "動作中"
    Label_Cond.Refresh
    For i = 0 To 256
        sts = WritePipe(hUSB, 2, DAdata(0), 2048, xfrlen)
    Next
    Label_Cond.Caption = "停止中"

End Sub

Private Sub Form_Load()
    Dim sts As Long
    Dim i As Integer
    start = 0
    For i = 0 To 16382 Step 2
        DAdata(i) = Sin(3.1416 * i / 1024) * 127 + 128  ' SIN(X)
        DAdata(i + 1) = 0
    Next

'GPIFウェーブフォームディスクリプタの設定
' Length/Branch :: Opcode ::  Output :: LogicFunction
    wsw(0) = 0: wsw(8) = 2: wsw(16) = &H3F: wsw(24) = 0
    wsw(1) = 0: wsw(9) = 2: wsw(17) = &H3F: wsw(25) = 0
    wsw(2) = 0: wsw(10) = 2: wsw(18) = &H3F: wsw(26) = 0
    wsw(3) = 0: wsw(11) = 2: wsw(19) = &H3F: wsw(27) = 0
    wsw(4) = 0: wsw(12) = 2: wsw(20) = &H3F: wsw(28) = 0
    wsw(5) = 0: wsw(13) = 2: wsw(21) = &H3F: wsw(29) = 0
    wsw(6) = 0: wsw(14) = 6: wsw(22) = &H3F: wsw(30) = 0
    wsw(7) = 0: wsw(15) = 0: wsw(23) = &H3F: wsw(31) = 0
    Call EZ_Open
    sts = EZ_SetPortConfig(2, 2, 1, 1, 1, 1, 1)
    sts = EZ_WaveSet(1, wsw(0))         ' 0:BRD 1:BWR 2:SRD 3:SWR
    sts = EZ_WaveSet(3, wsw(0))         ' 0:BRD 1:BWR 2:SRD 3:SWR
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Call EZ_Close
End Sub
```

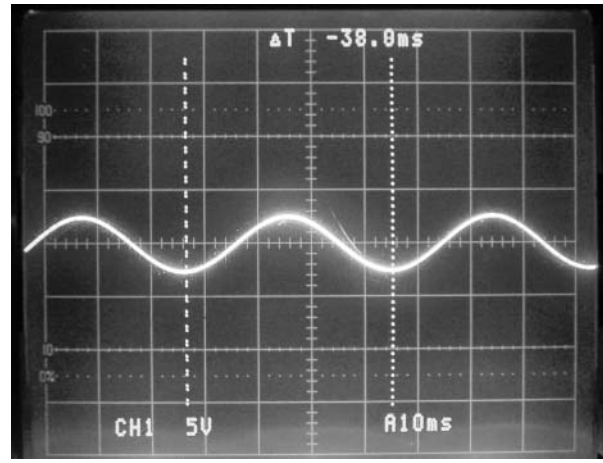

写真2　出力された正弦波データの波形

■GPIFモードのプログラム

VisualBasicのタイマ・コントロールでデータ更新を使う方法では，データ出力速度を上げるのは困難です．VisualBasicのタイマ・コントロールでは最小でも10ms程度がせいぜいというところなので，先ほどのように00h～FFhまでスイープするだけでも計算上2560ms，つまり2.6秒程度かかることになります．もっと高速なデータ出力を行うにはGPIFを使い，バルクOUTエンド・ポイントを使ってバースト転送を行うのが便利です．

プログラム（EzDACGPIF.FRM）の主要部を**リスト2**に示します（付属CD-ROMのEzDACGPIFフォルダに収録）．

●ウェーブフォーム・ディスクリプタの設計

高速伝送ができるGPIFですが，あまりにも速くデータを出力し終えてしまうと，ホスト側が追従できず，出力波形が途切れ途切れになってしまいます．今回はノンディジョン・ポイントを使って256×7＋1＝1793クロック（約37 μs）ごとに1ワード（2バイト）のデータを出力しています．USB2.0のハイスピード・モードで接続した場合には1パケットが512バイト（256ワード）で4パケットぶん，都合1024ワードぶんのデータがFX2側で蓄積可能ですので，約38 msぶんのデータが格納可能です．

ウェーブフォーム・ディスクリプタはすべてノンディシジョン・ポイントにしてみました．

LENGTH/BRANCH：00h（256クロック）
OPCODE：02h（データ出力），ステート6だけは06h（データ更新＋データ出力）
OUTPUT：3Fh（CTLは使っていないので何でも可）
LOGIC FUNCTION：00h（ノンディシジョン・ポイントでは未使用）

データの更新はステート6で行うようにしましたので，ステート6だけOPCODEフィールドのNEXTビットが‘1’になります．

EZ_SetPortConfig()でGPIFモードに設定してから，ウェーブフォーム・ディスクリプタを設定しておきます．

［START］ボタンが押されると，**EZ_GPIFTrig()**で出力カウンタをセットしてから，**WritePipe()**を使ってGPIFを使ったバースト出力を行わせます．

今回は2048バイト（1024ワード）単位のデータ送出を256回送出していますので，1/48［MHz］×（256 × 7 + 1）× 1024 × 256［μs］，すなわち約10秒間データが出力されます．

完成したら出力をオシロスコープなどで観察しながらSTARTボタンを押してみます．今回は出力データは正弦波データなので，正弦波が出力されているはずです．オシロスコープで見た波形は**写真2**のようになりました．

波形テーブルを操作すれば任意の波形を出力できるようになりますので，いろいろな波形出力を行わせてみるのも面白いでしょう．