応用編

第5章

アナログ信号の入力

…音声信号などの入力と光センサの実験

センサなどの信号はアナログ信号として検出されるものが普通です．一方，コンピュータで処理するためには数値化（ディジタル化）した情報でなくてはなりません．そこで，アナログ信号をディジタル化する目的で作られたのがA-Dコンバータ（Analog to Digital Converter）です．

A-Dコンバータは，その目的に応じて実にさまざまな方式が考え出されており，速度や分解能，外部インターフェースなどの工夫も含め，用途に応じて実にさまざまなICが作られています．

最も単純なA-D変換は，ある基準電圧以上か以下かで‘1’／‘0’を判定するものということになりますが，このままでは単なるスイッチ入力と変わりません．一般的には，ビデオ信号用の高速なA-Dコンバータで6ビット，汎用のもので8ビットのものが最小で，目的に応じて10ビット，12ビット…と分解能の高いICがラインアップされています．A-D変換には当然それなりの時間がかかります．速度，分解能，精度や価格などはトレードオフの関係にあって，どうしても高性能なものほど高いものになってきます．

5-1　使用するA-Dコンバータ IC

ここでは世の中に星の数ほどもあるA-DコンバータICのなかから，NS（ナショナル・セミコンダクター社）の汎用8ビットA-Dコンバータである ADC0804を選び，これをUCT-203ボードに接続してみることにしました．

型番の“ADC”の後に続く“08”は，8ビット分解能ということを示しているようです．それほど新しい製品ではありませんが，そのぶん枯れたICでもあり入手性も比較的よいのは利点です．今回はADC0804LCNというDIPパッケージ品を使いました．秋葉原ならば，若松通商などで入手できます．

●ADC0804の特徴

ADC0804は8ビットのA-Dコンバータで，ディジタル出力側は一般のCPU（マイコン）とインターフェースしやすいように考えられたものです．データシートを見ると，往年の8ビット・マイコンである8080マイクロプロセッサとダイレクト・インターフェースできるということがうたわれていますが，8080に限らず，一般的なワンチップ・マイコンなどとのインターフェースもしやすく，またCPUを使わずにスタンドアローンで動作させることもできる，なかなかよく考えられたA-DコンバータICであると言えるでしょう．

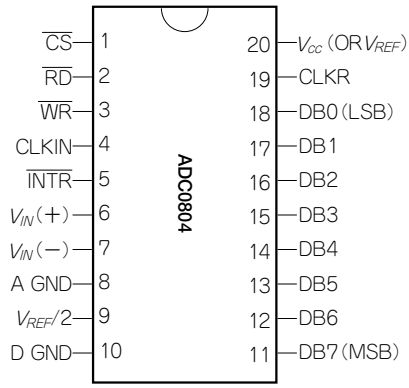

図1　A-Dコンバータ ADC0804 のピン配置

特徴を簡単にあげておきましょう．

(1) ＋5V単一電源動作（アナログ入力電圧0～5V）
(2) ロジック部の入出力電圧はTTL/CMOSとも可
(3) クロック・ジェネレータ内蔵（外部の*RC*で発振周波数を決定）
(4) ゼロ調整不要
(5) 分解能8ビット（精度±1ビット）
(6) 変換時間100 μs

●ADC0804の入出力信号

このICのピン配置は**図1**のようになっています．ADC0804LCNは，ごく一般的な20ピンのDIPパッケージです．

各信号ピンの意味を簡単に説明しておきましょう．詳細はナショナル・セミコンダクター社のサイトから日本語マニュアルがダウンロードできますので，そちらを参照してください．

▶$\overline{\text{CS}}$

Lアクティブのチップ・セレクト端子です．$\overline{\text{RD}}$信号や$\overline{\text{WR}}$信号とともに使われます．$\overline{\text{CS}}$端子がHレベルならば$\overline{\text{RD}}$信号，$\overline{\text{WR}}$信号とも無効です．

▶$\overline{\text{WR}}$

Lアクティブのライト信号です．A-Dコンバータなのに，なぜライト方向があるのかと思われるでしょう．実はライト動作は，ADC0804に対するA-D変換開始の指示になるのです．データがリードされたらA-D変換を開始するようにして，変換が完了したあとにデータを渡すという方法も考えられますが，ADC0804のデータ変換時間はクロックが640kHzのときに103～114 μsもかかります．いくら昔のCPUは遅かったとはいっても，これだけの時間待たされるのは良くないので，

① ライト動作（$\overline{\text{CS}}$＝$\overline{\text{WR}}$＝Lレベル）で変換を開始（データは任意）

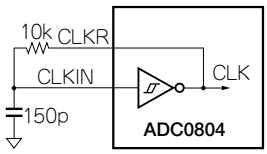

図2　*CR*による発振回路

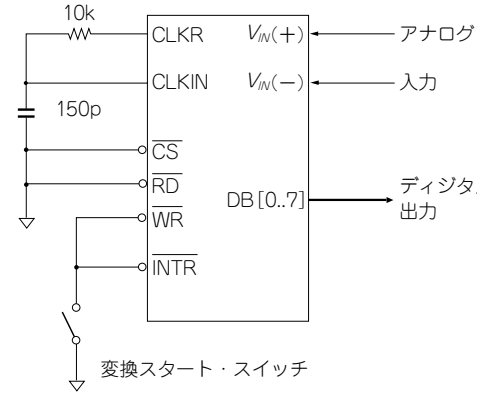

図3　フリーランニング・モードでの使いかた

② 変換が完了した時点でADC0804がCPUに割り込みをかける（$\overline{\text{INTR}}$ = Lレベル）

③ CPUが変換されたデータを引き取る

という3ステップでデータが渡されるようにしているのです．

$\overline{\text{WR}}$信号のパルス幅は最小100 nsが必要です．

▶$\overline{\text{RD}}$

Lアクティブのリード信号です．変換完了後のデータは$\overline{\text{CS}}$ = $\overline{\text{RD}}$ = Lレベルにすると，データ・バス（DB0～DB7）に現れます．データが出るまでの時間（アクセス・タイム）は最大200 nsです．

▶CLKINとCLKR

CLKINはクロック入力端子です．内部にはインバータがあって，このインバータの出力がCLKR端子から出ています．外部に発振回路がある場合にはCLIN端子から入力してもよいですし，CLKR端子を利用して*CR*発振回路を作って動作させることもできます．

今回は後者の*CR*発振回路を使いました．回路は**図2**のようになります．*R*が10 kΩ程度のときに，内部動作のクロック周波数f_{CLK}は，

$$f_{CLK}[\text{Hz}] = \frac{1}{1.1RC}$$

になります．*C*の値はデータシートを見ていると150 pFが標準のようですので，150 pFにしておきます．発振周波数の計算値は約610 kHzになります．

▶$\overline{\text{INTR}}$

Lアクティブの割り込み出力端子です．$\overline{\text{CS}}$と$\overline{\text{WR}}$によるライト動作によってA-D変換がスタートしたあと，変換動作が完了すると，この信号がアサート（Lレベルになる）されます．**図3**のように，$\overline{\text{CS}}$ = $\overline{\text{RD}}$ = Lレベルに固定して，$\overline{\text{INTR}}$端子を$\overline{\text{WR}}$端子と接続しておき，最初の1回だけ$\overline{\text{WR}}$端子をスイッチなどでLレベルに落とすようにすれば，常時A-D変換をしつづけるようにすることもできます．

今回はEzIO/FX2と接続してホスト（VisualBasicアプリケーション）からの要求で変換動作を行いますので，$\overline{\text{INTR}}$端子はUCT-203と接続しています．

▶ V_{IN}（+），V_{IN}（−）

信号電圧の入力端子です．差動入力になっていて，V_{IN}（+）とV_{IN}（−）の電圧差が変換されてディジタル・データになります．たとえば，入力電圧が1～5Vのとき，0Vから変換するようにすると，最小分解能は5÷256V（約20mV）ですが，V_{IN}（−）に1Vを与えることで，4÷256V（約16mV）にすることができます．また，差動で受けることで，信号に乗ってきたコモンモード・ノイズをキャンセルすることができます．

今回は簡単にV_{IN}（−）はGND（0V）に固定して使用し，V_{IN}（+）の電圧を変換するようにしました．

▶ AGND，DGND

グランド・ピンです．0V（GND）に接続します

▶ V_{CC}

電源ピンです．今回は，9Vの乾電池（006P）から3端子レギュレータを使って+5Vを作り出して使用しています．

▶ $V_{REF}/2$

A-D変換のための基準電圧です．ADC0804は，$V_{REF}/2$端子の電圧が変換電圧範囲の半分になるように動作します．たとえば，測定範囲を0.5V～3.5Vにしたい場合には，V_{IN}（−）端子は0.5Vに，$V_{REF}/2$端子は1.5V［(3.5 − 0.5)÷2］に設定します．

$V_{REF}/2$端子をオープンにしておくと，内部のプルアップ抵抗によってV_{CC}とGNDの中間の電圧になります．今回は電源電圧が+5Vで，0～5V入力にするので，オープンのままにしています．2.5Vのリファレンス電圧生成用ICなどを使って入力すれば，より高精度になりますが，今回のような用途ならば3端子レギュレータの精度程度でも充分でしょう．

▶ DB0～DB7

データ・バスです．$\overline{\text{CS}}$ = $\overline{\text{RD}}$ = Lレベルになると，DB0～DB7にA-D変換したデータが出力されます．

5-2　GPIFとのインターフェース

A-Dコンバータのコントロールは PIOモードで行うこともできますが，FX2に備わっているGPIFの汎用インターフェースとしての機能を生かすことから，GPIFを使うことにしました．接続関係は**図4**のようにしました．CTL信号で$\overline{\text{CS}}$端子，$\overline{\text{RD}}$端子，$\overline{\text{WR}}$端子の各信号を制御し，$\overline{\text{INTR}}$信号はGPIFで処理可能なRDY0端子につなぎます．FX2の入力は5Vトレラントなので，レベル・コンバータなどは特に使わず，直結で済ませました．

GPIFの利用によって，ホストから$\overline{\text{CS}}$や$\overline{\text{RD}}$，$\overline{\text{WR}}$などの信号線の細々した操作をせずに簡単に読み出すことができますし，連続したデータ転送も効率よく行うことができるようになります．

●回路

A-Dコンバータを接続する基板の回路を**図5**に，外観を**写真1**に示します．ADC0804には+5Vを供給したい

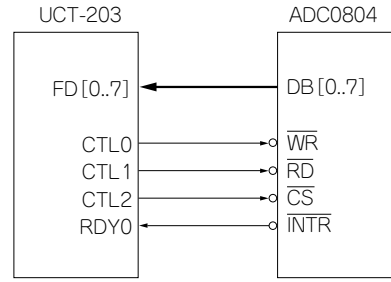

図4 UCT-203のI/Oコネクタとの接続

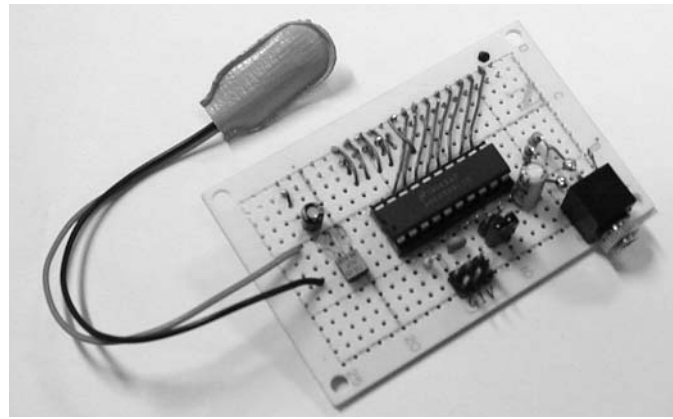

写真1 試作したA-Dコンバータ基板

ので，9Vの乾電池（006P）から，3端子レギュレータで＋5Vを作成しました．V_{REF}/2端子はオープンのままにしています．

入力端子は，ミニジャックを使ったAC入力（オーディオ入力を想定）と，ダイレクト入力の両方を用意して，ジャンパ・ピンで切り替えるようにしてみました．ダイレクト入力のほうは，ちょっとした外部回路をさらに増設できるように＋5Vも出力してみましたので，たとえば可変抵抗器を一つもってくればつまみの位置を読み取ることができます．

AC入力のほうは，無入力のときに入力電圧レンジ（0～5V）の中間である2.5Vになるように，1kΩの抵抗2本で2.5Vを作って与えています．

●ウェーブフォームの設計

先に少し触れたとおり，ADC0804をアクセスするときは，

① $\overline{\text{CS}}$ = $\overline{\text{WR}}$ = Lレベルにして変換スタート

② $\overline{\text{INTR}}$ = Lレベルによる変換完了通知を待つ

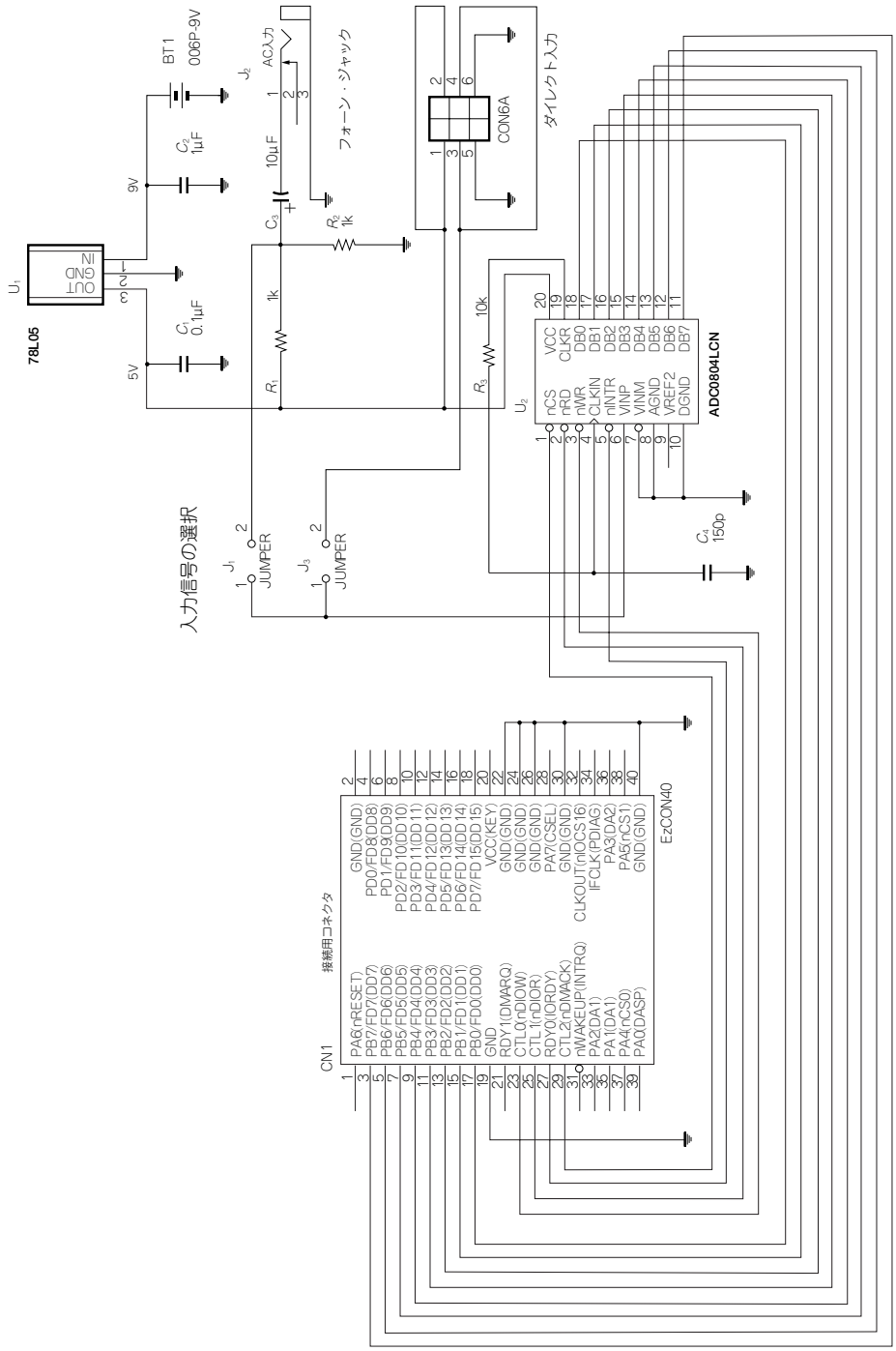

図5　A-Dコンバータ基板の回路

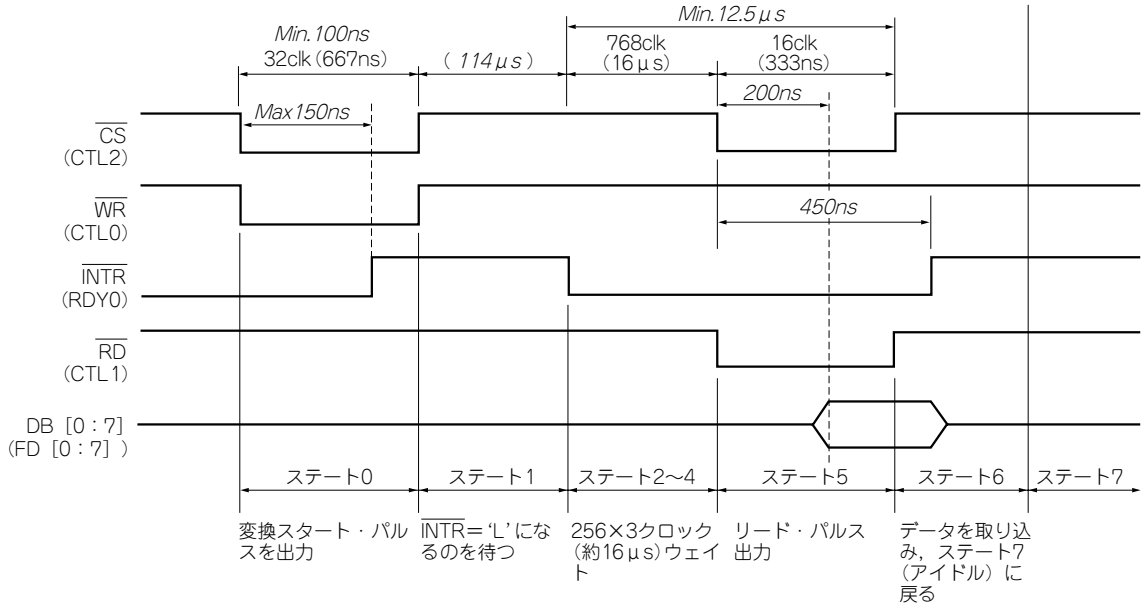

図6 ADC0804のアクセス・タイミングとGPIFのステート

③ $\overline{CS}$ = $\overline{RD}$ = Lレベルにしてデータ・バスに変換データを出力

という手順で動作させます．

一般のCPUを使った接続の場合には，それぞれを別々のオペレーションにするのが普通でしょう．もちろん，A-Dコンバータへのリード動作が行われたときに，ハードウェアで自動的に①から③までの動作が行われるように設計することもできますが，GPIFを使うとこの一連の動作を1回のアクセスとして設計してしまうことが簡単にできます．この方針で作ったのが**図6**のようなタイミングです．

図中で斜体になっているのが，ADC0804のデータシートに書かれているタイミング規定です．注意が必要なのは，ADC0804が$\overline{INTR}$信号をアサートしてから，$\overline{RD}$信号を立ち上げるまでの期間として最低8クロック（ADC0804のクロック周波数640 kHzとして12.5 μs）が必要とされています．この時間を確保するために，リード信号アサートまでの時間をノンディシジョン・ポイントを三つ使ってGPIFの768クロックぶん（1 ÷ 48 MHz × 768 = 16 μs）を確保しました．

それではステートの順を追って，簡単にウェーブフォームの説明をしておきましょう．

▶ステート0

$\overline{WR}$パルスを与えます．ADC0804は最小100 ns必要ということなので，100 ns以上あればよいのですが，$\overline{INTR}$端子がLレベル状態の場合，$\overline{WR}$の立ち下りから$\overline{INTR}$がHレベルに復帰するまでの時間が最大450 ns（300 $ns_{(typ.)}$）かかるということです．もし，$\overline{WR}$を100 ns程度で終わらせるとステート1でLレベルのチェックを

しても，$\overline{\text{INTR}}$がHレベルに復帰するまえの状態を取り込んでLレベルになったものと判断されてしまいます．

このため，$\overline{\text{INTR}}$がHレベルに戻るのに充分な幅だけステート0を延長します．ここでは余裕を見て32クロック（約667ns）を確保しました．

一定期間待つステートなので，ノンディシジョン・ポイントを使います．

▶ステート1

ADC0804の変換動作が完了し，$\overline{\text{INTR}}$がアサートされるのを待つステートです．$\overline{\text{INTR}}$はRDY0端子に繋ぎましたので，RDY0同士のAND演算をして（GPIFのディシジョン・ポイントでは必ず2入力間の演算が必要），‘1’ならば再びステート1に戻り，‘0’ならばステート2に移るようにします

▶ステート2～ステート4

先ほど触れたとおり，データシートにはADC0804が$\overline{\text{INTR}}$を出力しはじめてから，$\overline{\text{RD}}$が立ち上がるまでの時間は最低12.5 μs必要と書かれています．この時間を守らないと$\overline{\text{RD}}$による$\overline{\text{INTR}}$のリセット（Hレベルへの復帰）がうまくいかないことがあるということです．

GPIFは長いディレイを作るのは得意ではありません．ノンディシジョン・ポイントの1ステートでは最大256クロック（約5.3 μs）しか待てないので，今回はステートを三つ使って16 μsのウェイトを作りました．$\overline{\text{CS}}$と$\overline{\text{WR}}$はこのステートでHレベルに戻しておきます．

▶ステート5

リード・パルスを生成します．$\overline{\text{RD}}$がアサートされてからデータ制定まで200nsということですが，かなり余裕を見て16クロック（333ns）のパルス幅を作りました．これもノンディシジョン・ポイントで実現します．

▶ステート6

データの取り込み，$\overline{\text{RD}}$や$\overline{\text{CS}}$信号のネゲート（Hレベルへの復帰）を行い，ステート7（アイドル・ステート）に戻ります．ノンディシジョン・ポイントでもよいのですが，今回はディシジョン・ポイントを使ってステート7に飛ぶように書いてみました．

これを順次ステート・インストラクションに落としていけば完成です．実際のステート・インストラクションはソース・プログラム（EzADC.frm）の**Form_Load()**の中と，ステート・インストラクションの説明のページをつき合わせて見てください．

このウェーブフォームは，シングル・リード，バースト・リードのいずれでも使用可能です．

●アプリケーション・プログラムの作成

アプリケーションは**図7**のようなごく単純なものです．タイマ・コントロールを使い，タイマ・イベントの中で読み込んだデータを電圧値に変換して，数値として示すとともに電圧計のようなアナログ的な表示を行います．［ON/OFF］ボタンは，読み込み動作の開始/停止コントロールのためにつけてみました．

プログラムは**リスト1**（EzADC.frm）のようなごく単純なものです．**Form_Load()**のところでウェーブフォーム・ディスクリプタを設定しています．今回はシングル・リードしか使いませんが，バースト・リードでも同じように使えますので，バースト・リード側にも同じウェーブフォームを設定しています．手続きとしてはこれだけです．

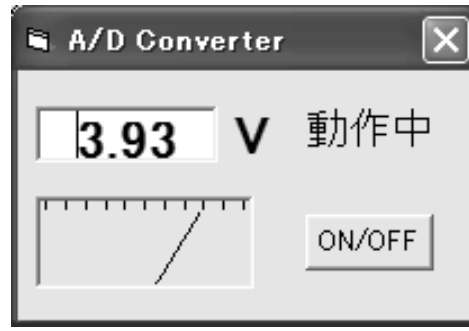

図7 A-Dコンバータ制御のプログラム

リスト1 A-Dコンバータの制御プログラムの主要部（EzADC.frm）

```
'================================================================================================
' Ａ－Ｄコンバータアクセスサンプル
'
'
' CTL0          -----> nWR
' CTL1          -----> nRD
' CTL2          -----> nCS
' RDY0          <----- nINTR
' PB(FD)[0..7] <====> DB[0..7]
'
'                 S7  |     S0         | S1 |   S2 S3 S4       |      S5         | S6->S7
'               _____|<-8cyc(167ns)->|____|<-768cyc(16us) ->|<-16cyc(333ns)->|_______
' CTL2(nCS)     _____¥________________/_____________________¥________________/_______
' CTL0(nWR)     _____¥________________/                                              ___
' RDY0(nINTR)___________________________¥_____________________/                    ___
' CTL1(nRD)     ___________________________________________¥_____________________/_______
'
'================================================================================================
Dim wsw(31) As Byte
Dim pastdat As Byte
Dim startadc As Byte

Private Sub Cmd_GetCVal_Click()
    If (startadc <> 0) Then
        startadc = 0
        Label_Active.Caption = "停止中"
    Else
        startadc = 1
        Label_Active.Caption = "動作中"
    End If
End Sub

Private Sub Form_Load()
    Dim sts As Long
' Length/Branch :: Opcode ::  Output :: LogicFunction
    wsw(0) = &H20: wsw(8) = 0: wsw(16) = &H3A: wsw(24) = 0
    wsw(1) = &HA: wsw(9) = 1: wsw(17) = &H3F: wsw(25) = 0
    wsw(2) = 0: wsw(10) = 0: wsw(18) = &H3F: wsw(26) = 0
    wsw(3) = 0: wsw(11) = 0: wsw(19) = &H3F: wsw(27) = 0
    wsw(4) = 0: wsw(12) = 0: wsw(20) = &H3F: wsw(28) = 0
    wsw(5) = &H10: wsw(13) = 0: wsw(21) = &H39: wsw(29) = 0
```

```
    wsw(6) = &H3F: wsw(14) = 3: 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, 0, 1, 0, 1)
    sts = EZ_WaveSet(0, wsw(0))        ' 0:BRD 1:BWR 2:SRD 3:SWR
    sts = EZ_WaveSet(2, wsw(0))        ' 0:BRD 1:BWR 2:SRD 3:SWR
    startadc = 0
    Call Cmd_GetCVal_Click
End Sub

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

Private Sub Timer1_Timer()
    Dim Analog_val As Long
    Dim i As Integer
    For i = 0 To 5
        Picture1.Line (Picture1.Width * 0.9 / 10 * i + Picture1.Width * 0.95 / 2,
                                                        Picture1.Height * 0.01)
                    -(Picture1.Width * 0.9 / 10 * i + Picture1.Width * 0.95 / 2,
                                                        Picture1.Height * 0.1)
        Picture1.Line (-Picture1.Width * 0.9 / 10 * i + Picture1.Width * 0.95 / 2,
                                                        Picture1.Height * 0.01)
                    -(-Picture1.Width * 0.9 / 10 * i + Picture1.Width * 0.95 / 2,
                                                        Picture1.Height * 0.1)
    Next
    Picture1.Line (Picture1.Width * 0.95 / 2, Picture1.Height * 1.1)
                -((pastdat - 128) / 256 * Picture1.Width * 0.9
                          + Picture1.Width * 0.95 / 2, Picture1.Height * 0.1),
                                                          Picture1.BackColor
    If (startadc <> 0) Then
        Analog_val = EZ_SglRd(0) And &HFF
        pastdat = Analog_val
        Text_AVal.Text = Format(Int((5# * Analog_val / 256) * 100 + 0.5) / 100,
                                                                    "fixed")
        Picture1.Line (Picture1.Width * 0.95 / 2, Picture1.Height * 1.1)
                    -((pastdat - 128) / 256 * Picture1.Width * 0.9
                           + Picture1.Width * 0.95 / 2, Picture1.Height * 0.1),
                                                              RGB(0, 0, 0)
    Else
        Picture1.Line (Picture1.Width * 0.95 / 2, Picture1.Height * 1.1)
                    -((pastdat - 128) / 256 * Picture1.Width * 0.9
                           + Picture1.Width * 0.95 / 2, Picture1.Height * 0.1),
                                                            RGB(255, 0, 255)
    End If
End Sub
```

Timer1_Timer()の中での実際のデータ読み込みは，

Analog_val = EZ_SglRd(0) And &HFF

という1行です．A-D変換の開始，完了待ち，データ読み込みという一連の操作は，すべてGPIFが受け持つので，ホストからは単なるPIOリードと同じ感覚でアクセスできてしまいます．

もちろん，**EZ_GPIFTrig()**を使って，バースト・リードをかければ自動的に連続リードが行われ，データが取得されます．ただ，ここで問題なのは，ADC0804の変換時間が常に一定とは限らないということです．通常，

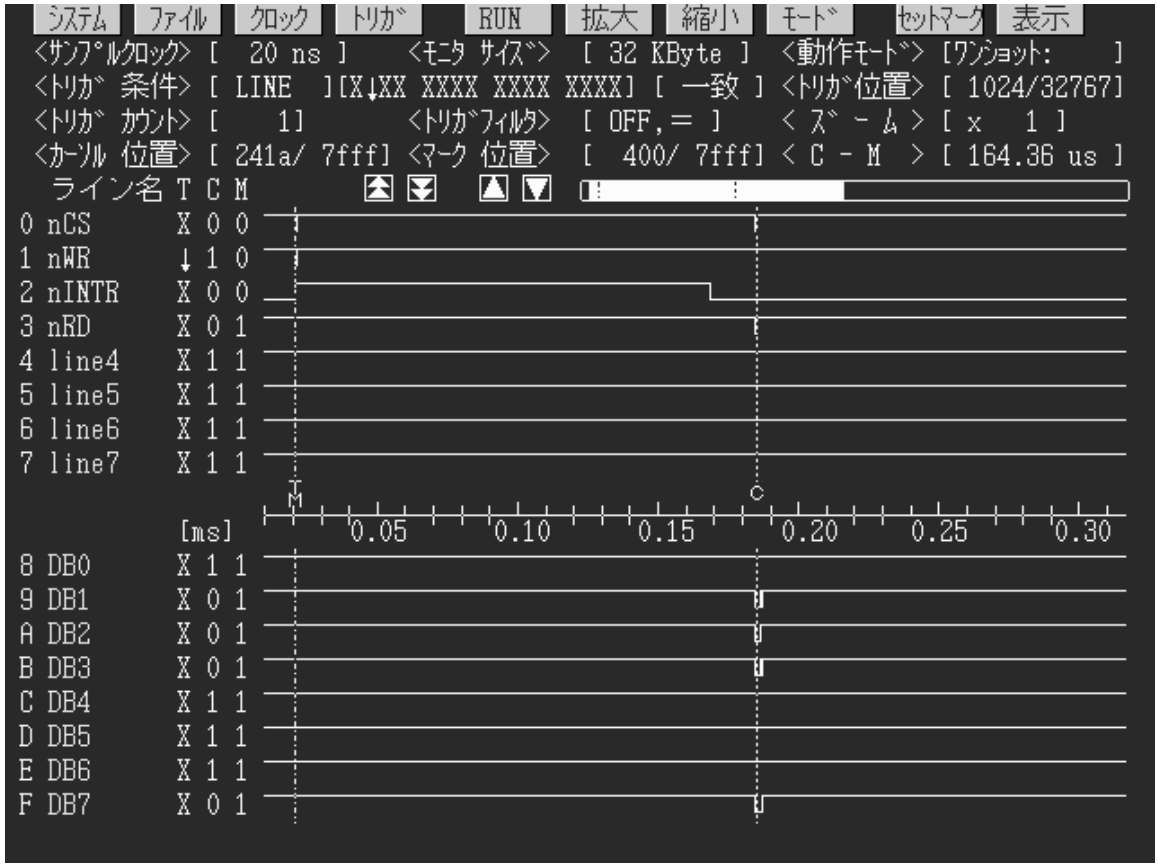

図8 1回のアクセス動作

バースト的にデータを取りたい場合，一定間隔で取り込みたいことがほとんどですが，今回の方法ではADC0804の変換完了を待ってから次に進むため，変換時間によるばらつきが出てしまうのです．外部にハードウェアでタイマを設けたり，$\overline{\text{INTR}}$が必ずアサートされるだけの時間を固定でウェイトさせるなどの方法もありますが，外部でタイミングを取る回路が必要になることから，今回は見送りました．

興味のある方はいろいろと実験されてみると面白いでしょう．

●実行例

回路基板が製作できたら，電源まわりをもう一度チェックしておきます．特に乾電池から＋5Vを作っている3端子レギュレータの周辺は厳重にチェックしてください．ここで間違えるとADC0804だけではなく，UCT-203ボード側まで壊しかねません．できれば，ADC0804はソケット付けにして，UCT-203とも繋がずに電池をつなぎ，電源電圧だけでもチェックしておくとよいでしょう．問題がなければ，ADC0804をセットして，UCT-203ボードとつなぎ，動かしてみます．

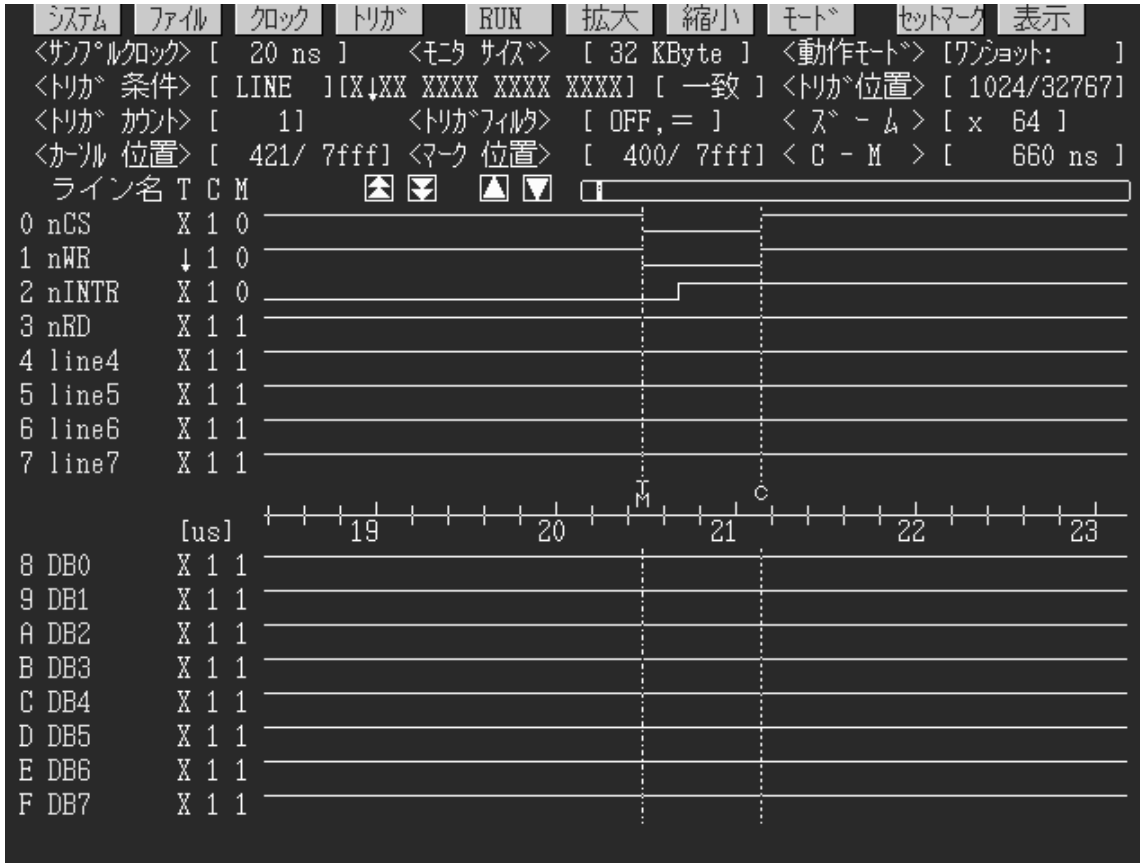

図9　最初のライト動作部分を拡大

入力信号を設定するジャンパは，とりあえずAC入力側にしておきましょう．アプリケーションを起動すると，約2.5Vを表示するはずです．ここでフォーン・ジャックにラジカセの出力などを繋いで音を入れれば，VUメータのように音声の大きさに合わせてメータが振れ，表示がパラパラと動きます．

それでは念のため動作波形を見ておきましょう．**図8**が1回のアクセス動作を示したものです．最初のライトが行われて$\overline{\text{INTR}}$（図ではnINTRというライン名）がHレベルになり，変換完了で$\overline{\text{INTR}}$がLレベルになります．この後リードされてデータが出てくるという，一連の動作が予定どおり行われていることが読み取れます．

最初のライト動作部分を拡大したのが**図9**です．$\overline{\text{CS}}$（図ではnCS）と$\overline{\text{WR}}$（図ではnWR）によるライト・パルスの幅は約660nsあり，計算どおりというところです．$\overline{\text{WR}}$が立ち上がり，GPIFがステート1に移るよりも充分まえに$\overline{\text{INTR}}$信号がHレベルに復帰していることが見て取れます．

このあと，$\overline{\text{INTR}}$がLレベルになってからリードが行われるまでの波形が**図10**です．約16 μsですから，こちらもほぼ予定どおりです．

実際にデータを読み出しているリード・タイミングが**図11**です．サンプリングが20 nsですので，計算値333 ns

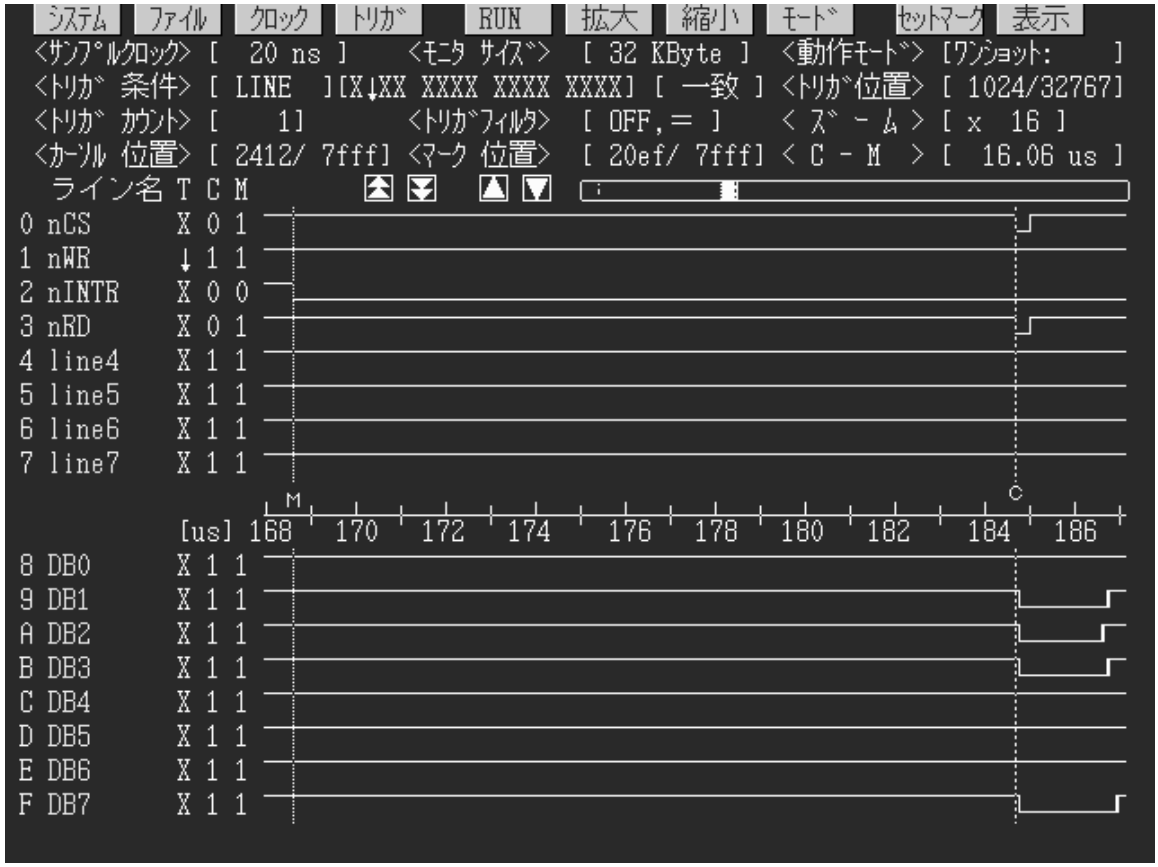

図10　/INTRがLレベルになってからリードが行われるまでの波形

に対して340 nsのリード・パルス幅として計測されていますが，問題はありません．データは充分に早く出ていますし，データ・ホールド・タイムも充分あるので，データ取り込みでタイミング・マージンが不足することはないと考えてよいでしょう．

5-3　フォト・センサの接続

せっかくA-Dコンバータが接続できたのですから，簡単な計測らしきことをやらせてみることにしましょう．何か適当なセンサということで，ここでは明るさの検出をやらせてみました．世の中にはいろいろなセンサがありますが，同じように0～5V程度の電圧変化に変換できれば取り込み可能です．

■CdSセルを使ったフォト・センサ

CdSセル（硫化カドミウム・セル：略してカドミウム・セル）は光が当たると抵抗値が大きく下がるデバイス

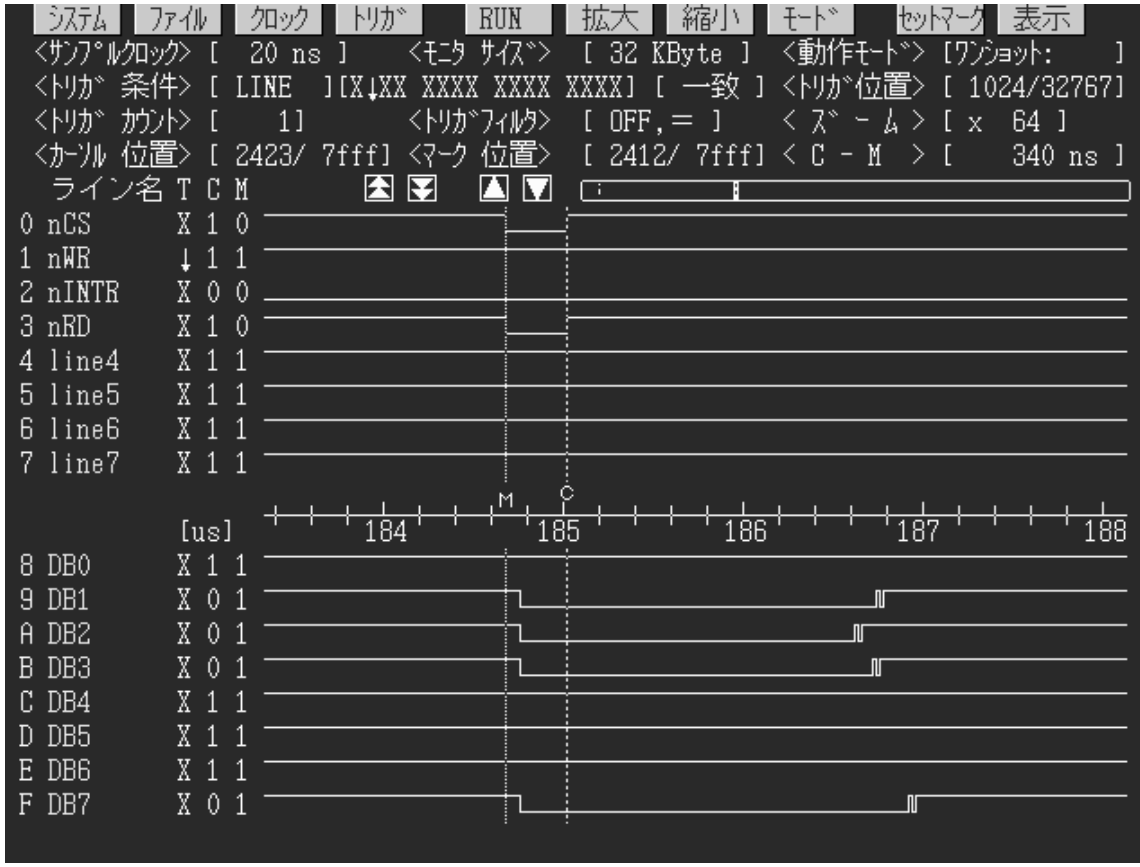

図11　データを読み出しているリード・タイミング

です．応答速度は遅いので，光を使ったデータ転送などには不向きですが，構造が単純で簡単に作れることから街灯の自動点灯/消灯などに使われてきました．また，応答が遅いとはいってもある程度の速度はもっていますので，おもちゃの光線電話などにも利用されていました．

今回使ったCdSセルは直径が2cmほどのものです．光が当たらないと数MΩ程度と，かなり高い抵抗値を示しますが，室内照明の明るさで2kΩ程度まで抵抗値が下がります．

今回製作したA-Dコンバータ基板のダイレクト入力側を利用してインターフェースした回路が**図12**，外観が**写真2**の下側です．単純に，CdSセルと抵抗で＋5Vの電圧を分圧して得られた電圧を読み取ろうというものです．

●実行例

できあがったら早速つないでみましょう．A-Dコンバータのサンプル・プログラムを起動して電圧を読み取りながら，CdSセルを手で覆うと値が小さくなり，照明器具のほうに向けると値が大きくなるはずです．コネクタを逆向きに入れると，今度は明るいほど値が小さくなるようにすることもできます．今回は抵抗値を固定値にし

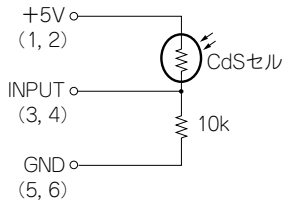

図12　CdSセルを使った光センサの回路

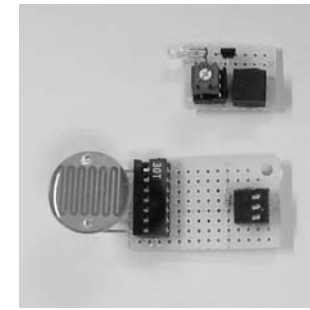

写真2　試作したフォト・センサ基板の外観

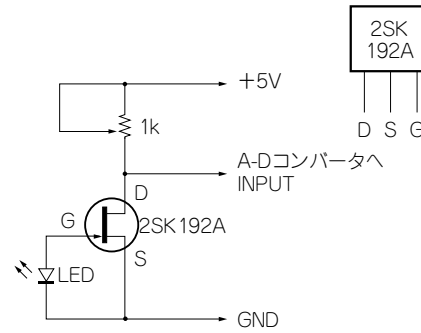

図13　一般のLEDを使った光センサ

ましたが，検出したい明るさの範囲によって抵抗値を変えてみるとよいでしょう．

CdSセルは極性もありませんので扱いが非常に楽である反面，カドミウムを使用しているということや，応答速度が遅いことなどのデメリットがあり，最近ではフォト・トランジスタなどが使われることが一般的になってきています．

■LEDを使ったフォト・センサ

LEDは電流を流して発光させるのが本来の使いかたですが，実は外部から光を当てるとアノード側にプラス，カソード側にマイナスの電圧が発生します．実はLEDに限らず，通常のダイオードであっても同じように光を当てると電圧が発生します．いわゆるフォト・ダイオードでは，この現象がより強く現れるようにしているのです．

もちろん，フォト・ダイオードを利用してもよいのですが，今回はちょっと面白いということと，100円ショップでも入手できるということで，試しにLEDを使ってみることにしました．

LEDで発生する電圧はそれなりにあるのですが，電力が非常に小さいので，ちょっとした抵抗でもすぐ電圧が落ちてしまいます．このままではA-Dコンバータに入力するのは無理なので，FETを使ったバッファを設けます．CdSのときと同様にダイレクト入力側の端子を使って，**図13**のような回路を接続します．

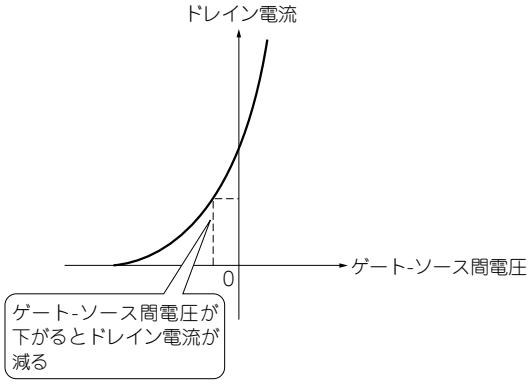

図14　2SK192Aの特性と利用

光が当たっていないときはLEDは単なるダイオードですので，ソース（S）とゲート（G）の電圧は同じです．2SK192Aはゲート電圧がゼロのときにもドレイン電流が流れますので，抵抗を使って電圧降下ぶんを測定します．2SK192Aの特性は**図14**のようになっていますので，ゲート-ソース間電圧がゼロでもドレイン電流は流れます．

ここで，LEDに光が当たると，ゲート電圧がソース電圧よりも低くなり，ドレイン電流が減ってきます．つまり，電圧降下ぶんが減る（ドレイン端子の電圧が上昇する）のです．

この変化をA-Dコンバータで測定するという仕組みです．

●製作と実験

回路はごく単純なものなので，ごく小さく収まってしまいます（**写真2**の上側）．間違えないように組み立てたら，とりあえず半固定抵抗を真ん中あたりにして先に作っておいたA-Dコンバータ基板に接続してみます．LEDを手で覆ったり，光を当てると値が変わることでしょう．

抵抗値を変えてみたり，回路をもう少し工夫してみれば，さらに良い結果が得られると思います．