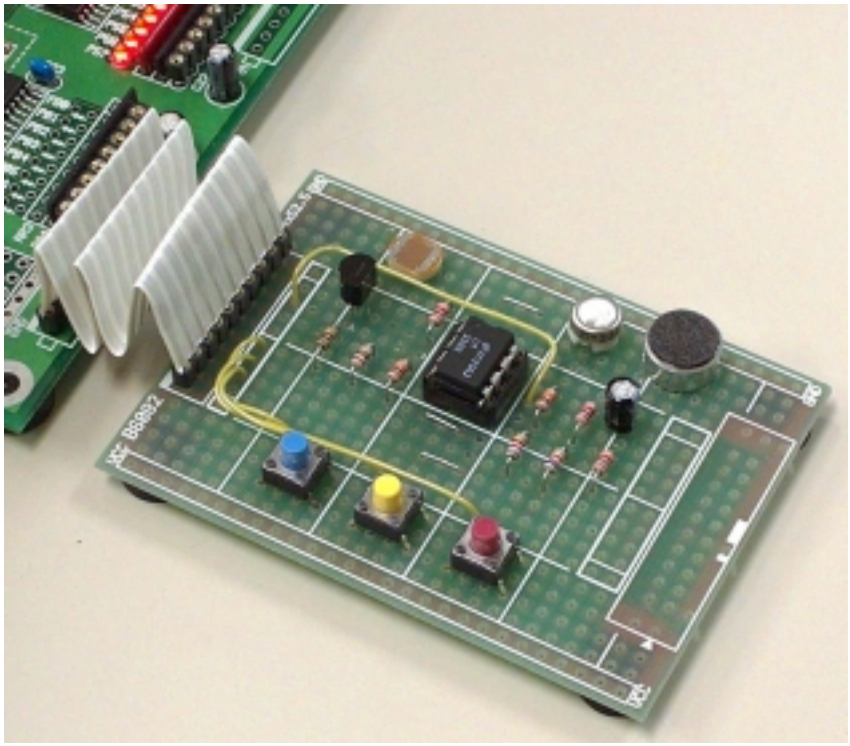
－TK-3687 / TK-3687mini Option－

AD コンバータ

Version 2.05



目　次

Toyo-linx,Co.,Ltd.

1 はじめに、、、

自然界の物理量、例えば、温度、湿度、重さ、音声等は全てアナログ量です。一方、マイコンはデジタル値しか取り扱うことができません。ということは、マイコンで物理量を取り扱うためにはアナログ値を何らかの方法でデジタル値に変換する必要があります。このような働きをするデバイスが A/D コンバータです。

H8/3687 の A/D コンバータは 10bit の分解能を持ってます。これはどれくらいの分解能かといいますと 0～5V の電圧を 1024 ステップに分解することができます。つまり 1 ステップあたりの電圧変化量は 5V を 1024 ステップでわるので約 0.005V(5mV)となります。また、変換時間も 1 チャンネルあたり最小 3.5μsec と高機能な A/D コンバータです。

A/D コンバータに入力できるのは電圧だけなので、物理量を扱う際にはセンサによって電圧へ変化する必要があります。このキットでは H8/3687 内蔵の A/D コンバータを用いて明るさと音の二つのアナログ値をデジタル値に変換します。明るさを電圧へ変化させるセンサに CdS を、音を電圧へ変化させるセンサにコンデンサマイクをそれぞれ用います。

学習する内容は 2 つです。まず、光センサを用いて H8/3687 の A/D コンバータの基本的な使い方と平均化の方法を学習します。次に、コンデンサマイクを用いて簡易レベルメータをプログラムします。光センサとは違い音は交流的な信号の変換ですから、変換された値の取り扱い方が多少違っています。最後に応用としてマイクに入力された音声の大きさを比較する最大音量計のプログラムを紹介します。

作業の流れは、、、

ハードの組み立て
↓
光センサの A/D 変換プログラム
↓
簡易レベルメータのプログラム
↓
最大音量計

の順で説明していきます。

作業を進めていくに当たり次のものを用意して下さい。

・組み立て

ハンダごて、ハンダ、ニッパ、ワイヤストリッパ、ピンセット

・プログラム

High-performance Embedded Workshop(HEW)、もしくは Motorola 形式 HEX ファイル(*.mot)を生成するアセンブラ(本文内で掲載しているリストは HEW を使用して作成しています)

・動作確認

ハイパーターミナル（Windows 付属）、Excel&VBA による表示プログラム（CD 在中）、D-Sub9pin ストレートケーブル

TK-3687mini で使用される方は巻末の「TK-3687mini で使うためのハードの組立て」をご覧下さい。なお，プログラムは共通して使用できます。

2 ハードの組み立て

各プログラムを作成する前に付属のユニバーサル基板にハードを組み上げます。プリント基板と違いユニバーサル基板は全ての配線を自分で結線しなければなりません。P.4 の回路図を見ながら部品をハンダ付けしていく事によって、ハードの構成を理解してみましょう。

まず、工具・部品の確認を行いましょう。下記の部品表と照らし合わせて全ての部品が揃っている事を確認して下さい。

■工具

ハンダごて、ハンダ、ニッパ、ワイヤストリッパ、ピンセット

■部品

TK-3687オプション ADコンバータ 部品表	全数:	1

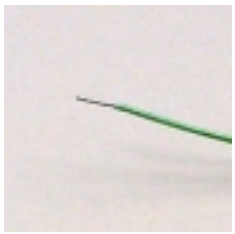
	部品番号	型名, 規格	メーカー	数量	全数	備考
1	OP1	LM358N		1	1	単電源デュアルオペアンプ
2	Q1	2SC1815		1	1	
3	CdS			1	1	光センサ
4	ECM			1	1	コンデンサマイク
5	R9	100Ω		1	1	
6	R8	200Ω		1	1	
7	R1,2,4,6	1kΩ		4	4	
8	R3	620Ω		1	1	
9	R5	820Ω		1	1	
10	R7	10kΩ		1	1	
11	VR1	100kΩ		1	1	
12	C2	0.1μF (積セラ)		1	1	
13	C1	10μF/16V (電解)		1	1	
14	SW1,2,3			3	3	
15	CN1,B	10pin 丸ピンソケット		2	2	20pinを2つに切断 ※1
16	基板	B6082	東洋リンクス	1	1	
17	ゴム足			4	4	
18	ラッピングケーブル	50cm		1	1	結線用
19	メッキ線					Vcc,GND等半田面結線用 ※2
20						
21	接続ケーブル	A10-N100-A	Hitaltech	1	1	CN-B接続用
22						

※相当品を使用する場合があります。
※1 1つはTK-3687の右辺下のCN-Bに実装します。
※2 ラッピングケーブルの被覆を剥し2本をよじって使用します。また抵抗やコンデンサの足も流用できます。

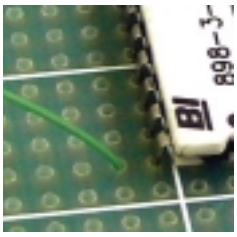
部品の確認が済みましたら、いよいよ組み立てです。4 ページの回路図・実装図をよく見ながらハンダ付けを行って下さい。ハンダ付けによるケーブル配線は大変なので、電源や GND、交差しない信号線等はハンダ面でメッキ線や部品の余ったリード線を流用して接続します(P.5 完成写真を参照)。ハンダ面のみでは配線しきれない信号線はラッピングケーブルで配線していきます。尚、ラッピングケーブルでの配線が初めての方は次ページの結線方法を参照して下さい。

■ラッピングケーブルでの結線方法

ハンダ面で結線されない配線はラッピングケーブルで結線していきます。ここではラッピングケーブルでの結線の仕方を説明します。

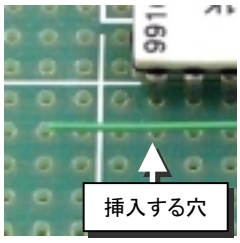


1.
被覆を剥きます
(5mm 位)



2.
部品面からに穴に通します

3.
ハンダ付けする部品の足にラッピングケーブルを絡げてからハンダ付けします



4.
結線先までケーブルを這わせ、挿入する穴から 3 つ先の穴辺りでカットします
1～3 の手順でハンダ付けします

★コンデンサマイクの極性について

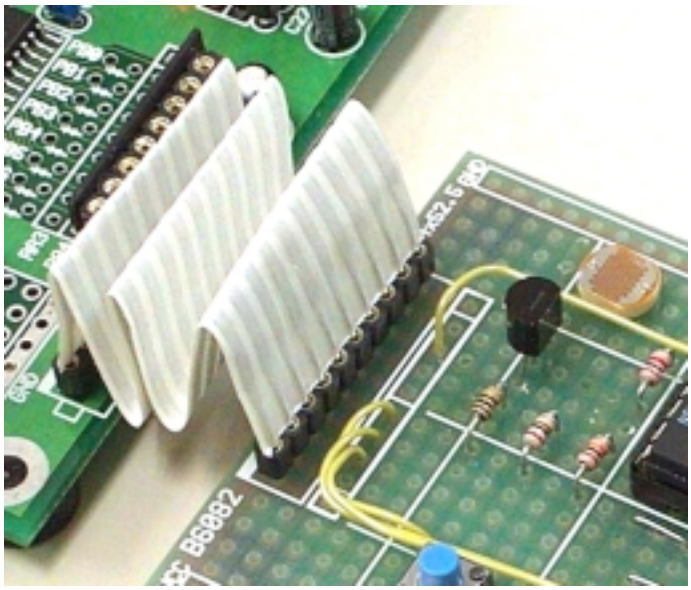
コンデンサマイクには極性があります。裏面（リードがついている方）から見て基板が黒く塗られている方が“－”、塗られていない方が“＋”になります。逆に取り付けた状態で電源を投入すると壊れますので絶対に間違えないように注意してください。実装してから極性が分かるように油性ペン等で印をつけておくと良いでしょう。

黒い方が
“－”

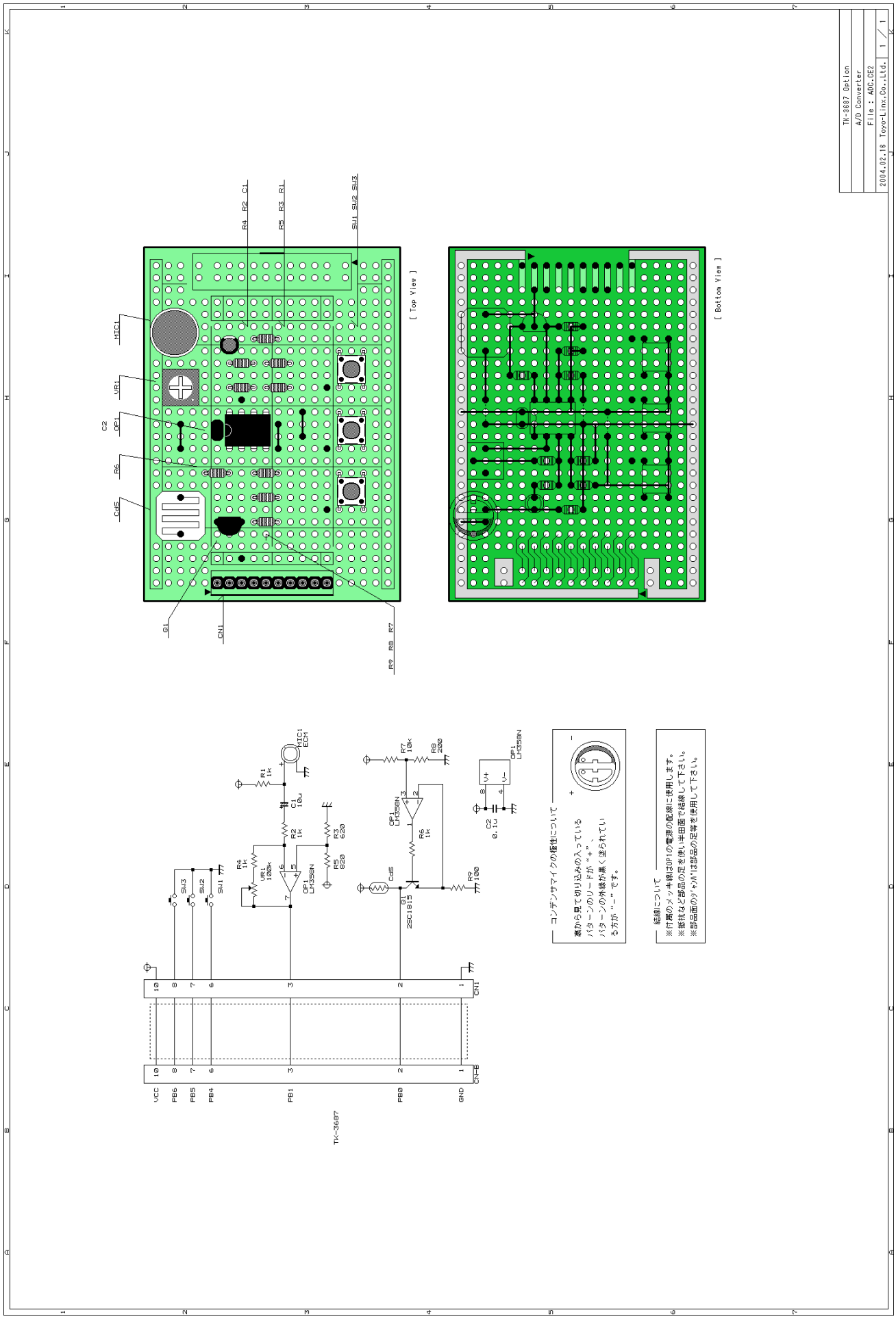
実装しても分かるよう
“－”側に印をつける

■TK-3687 との接続

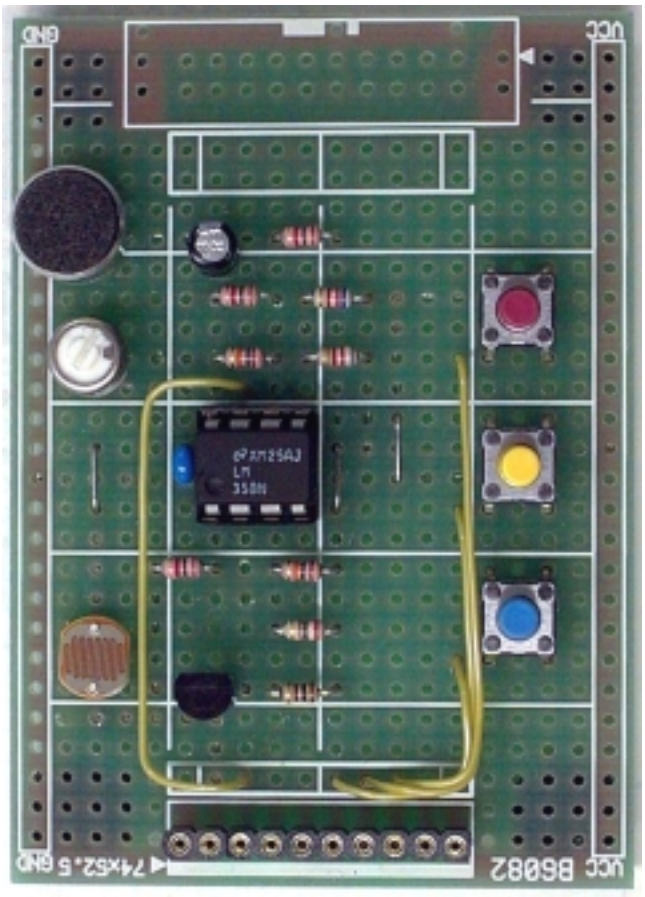
全てのハンダ付けが終了したら TK-3687 と接続しますが、最後にもう一度結線に間違えが無いかよく確認して下さい。間違えが無いようでしたら、10 芯のフラットケーブルで TK-3687 の CN-B と基板とを接続します。芯が折れ曲がらないように気を付けながらしっかりと挿入して下さい。

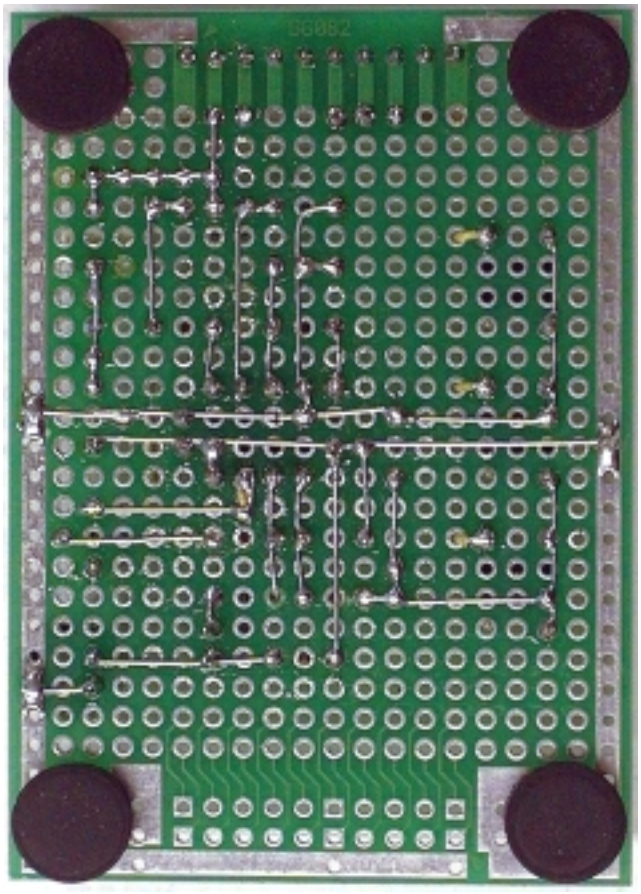


■回路図・実装図


TK-3687 Option
A/D Converter
File : A0C.CE2
2004.02.16 Toyo-Linx,Co.,Ltd.
[Top View]
[Bottom View]
コンデンサマイクの極性について
裏から見て切り込みの入っている
パターンのリードが"+"、
パターンの外縁が黒く塗られてい
る方が"−"です。
結線について
※汎用のメッキ線はDIPの電源の配線に使用します。
※抵抗など部品の足を使い半田面で結線してください。
※部品面のｼﾞｬﾝﾊﾟ部品の足等を使用して下さい。
TK-3687

■完成写真





■動作チェック

間違えが無いようなら動作確認を行います。パソコンと TK-3687 を接続して内蔵の簡易モニタを起動します。モニタでファイルのロードと実行のコマンド“LG”を入力し、CD-ROM に収録されているプログラムをロード・実行しチェックを行います。ファイルの場所は、E:¥TK-3687¥ｵﾌﾟｼｮﾝ¥AD ｺﾝﾊﾞｰﾀ¥ﾌﾟﾛｸﾞﾗﾑ（CD-ROM ドライブが E の場合）です。

最初は光センサ部のチェックです。プログラム“AD_send.mot”をロードし実行してください。ハイパーターミナルの画面に 3 桁の数字が表示されていきます。表示されている値が現在の明るさを示していますので、手などで光センサを覆うなどして明るさを変化させてください。表示されている値が暗くすれば小さく、明るくすれば大きくなれば正常に動作しています。もし変化しない場合は、もう一度ハンダ付けや抵抗の定数、トランジスタの等をチェックしてください。

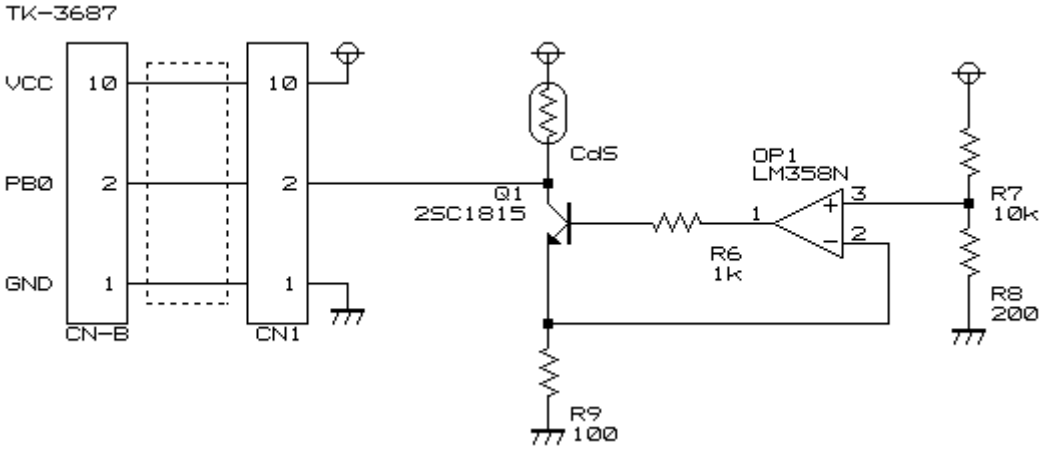
次にコンデンサマイク部のチェックを行います。プログラム“AD_Lvmeter01.mot”をロードし実行してください。コンデンサマイクを指で軽くたたくとそれに合わせてポート 5（以降 P5）の LED が点灯するはずです。ボリュームを左に回すと LED の点灯する振れ幅が小さく、また右に回すと大きく振れれば OK です。LED が何も反応を示さない時はもう一度ハンダ付けのチェック、コンデンサマイクの極性、抵抗の定数等をチェックしてください。

3 光センサの A/D 変換プログラム

この章では明るさというアナログ量を CdS（光センサ）で電圧に変換、A/D 変換しその結果を 0～255 までの 10 進数でハイパーターミナルの画面へ表示させます。表示間隔は約 1 秒です。ここでは H8/3687 の A/D のレジスタ等基本的な変換の方法と平均化について学習します。なお、ハイパーターミナルへの送信部“SCI3 のイニシャライズと送信”については巻末の付録を参照して下さい。

■ハード

図 3-1 は光センサ部を抜き出した回路図です。オペアンプとトランジスタで定電流回路を構成しています。CdS（光センサ）は明るさによって抵抗値が変化する素子です。CdS の抵抗値に関らず流れる電流は一定ですので抵抗値の変化はそのまま電圧の変化になります。得られた電圧（＝アナログ信号）は TK-3687 の PB0 に入力されています。



＜図 3-1 光センサ部回路図＞

■ソフト

次に A/D 変換で使用するレジスタの説明をします。使用するレジスタは次の 2 つです。

• ADCSR（A/D コントロール／ステータスレジスタ）

A/D の制御、ステータスを表します

7	6	5	4	3	2	1	0
ADF	**ADIE**	**ADST**	**SCAN**	**CKS**	**CH2**	**CH1**	**CH0**

・bit7 ADF ・・・ A/D エンドフラグ
変換が終了すると 1 がセットされます
・bit6 ADIE ・・・ A/D インタラプトイネーブル
1 にセットすると割り込み要求がイネーブルになります
・bit5 ADST ・・・ A/D スタート
1 にセットすると変換を開始します
・bit4 SCAN ・・・ スキャンモード
A/D 変換のモードを選択します “0”＝単一モード ／ “1”＝スキャンモード
・bit3 CKS ・・・ クロックセレクト
A/D 変換時間を設定します “0”＝134 ステート(max) ／ “1”＝70 ステート(max)
・bit2-0 CH2-0 ・・・ チャンネルセレクト
アナログ入力チャンネルを選択します

CH2	CH1	CH0	SCAN=0	SCAN=1
0	0	0	AN0	AN0
0	0	1	AN1	AN0～AN1
0	1	0	AN2	AN0～AN2
0	1	1	AN3	AN0～AN3
1	0	0	AN4	AN4
1	0	1	AN5	AN4～AN5
1	1	0	AN6	AN4～AN6
1	1	1	AN7	An4～AN7

・ADDRA～D（A/D データレジスタ A～D）

A/D 変換結果を格納する 16bit レジスタです

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
変換結果(10bit)										**0**	**0**	**0**	**0**	**0**	**0**

アナログ入力チャンネルと A/D データレジスタの対応

アナログ入力チャンネル		変換結果が格納される A/D データレジスタ
グループ 0	グループ 1	
AN0	**AN4**	**ADDRA**
AN1	**AN5**	**ADDRB**
AN2	**AN6**	**ADDRC**
AN3	**AN7**	**ADDRD**

今回は AN0 のみ使用して A/D 変換を行いますので単一モード(SCAN=“0”)で動作させます。光の変化ですのでそんなに変換スピードは要求されません。ですので変換時間は 134 ステート(CKS =“0”)とします。CH0-2 は AN0 なので “000”、A/D データレジスタは ADDRA を使用することになります。

サンプルプログラムとして“AD.src”をリスト 3-1 に示します。このプログラムは 1 回の A/D 変換を行い汎用レジスタ R0 の bit0～9 に A/D 値をセットし、その A/D 値をポート 5 及びポート 1 にバイナリーで出力する A/D 変換の基本となる部分です。A/D10bit の内、bit0-7 をポート 5、bit8-9 をポート 1 に出力します。尚、この後に作るプログラムは平均化処理を加えたり、タイマやパソコンへの送信を追加してより機能アップしていきますが、A/D 変換の考え方は同じです。従って、ここがスタートラインですから、その考え方をしっかりマスタしなければなりません。

```
;---------------------------------------------------------------------
;                                                                     |
;   FILE        :AD.src                                               |
;   DATE        :Mon, Feb 16, 2004                                    |
;   DESCRIPTION :Main Program                                         |
;   CPU TYPE    :H8/3687                                              |
;                                                                     |
;   This file is generated by Hitachi Project Generator (Ver.2.1).    |
;               programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.         |
;                                                                     |
;---------------------------------------------------------------------
; ■ 概要
;   ﾁｬﾝﾈﾙ AN0 の A/D 変換を行いその結果をﾎﾟｰﾄに出力します。変換結果の bit7-0 は
; ﾎﾟｰﾄ 5 に、bit9-8 はﾎﾟｰﾄ 1 に出力します。

;-----  ｲﾝｸﾙｰﾄﾞﾌｧｲﾙ  ----------------------------------------------------
     .include  "io3687F_equ.inc"    ;H8/3687F 内蔵ﾓｼﾞｭｰﾙ ｱﾄﾞﾚｽ定義
```

＜リスト 3-1　AD.src （1/2）＞

```
     .export   _main
_main:
;-----  PIO ｲﾆｼｬﾗｲｽﾞ  --------------
     mov.b     #H'FF,r0l
     mov.b     r0l,@PCR1           ;ﾎﾟｰﾄ 1 出力
     mov.b     r0l,@PCR5           ;ﾎﾟｰﾄ 5 出力

;-----  A/D ｲﾆｼｬﾗｲｽﾞ  --------------
     mov.b     #B'00000000,r0l     ;単一ﾓｰﾄﾞ AN0 134 ｽﾃｰﾄ変換
     mov.b     r0l,@ADCSR

;-----  A/D 変換  -------------------
AD_00:
     bset      #5,@ADCSR           ;変換開始
AD_02:
     btst      #7,@ADCSR           ;変換終了 ?
     beq       AD_02
     bclr      #7,@ADCSR           ;変換終了

     mov.w     @ADDRA,r0           ;A/D 値ﾘｰﾄﾞ
     shlr.w    r0                  ;ﾃﾞｰﾀ右詰め
     shlr.w    r0                  ; bit15-6 → bit9-0
     shlr.w    r0
     shlr.w    r0
     shlr.w    r0
     shlr.w    r0

;-----  ﾎﾟｰﾄへ出力  ----------------
     mov.b     r0l,@PDR5           ;ﾎﾟｰﾄ 5 = A/D 値[bit7-0]
     mov.b     r0h,@PDR1           ;ﾎﾟｰﾄ 1 = A/D 値[bit9-8]

;---------------------------------
     bra       AD_00

;=====================================================================
     .end
```

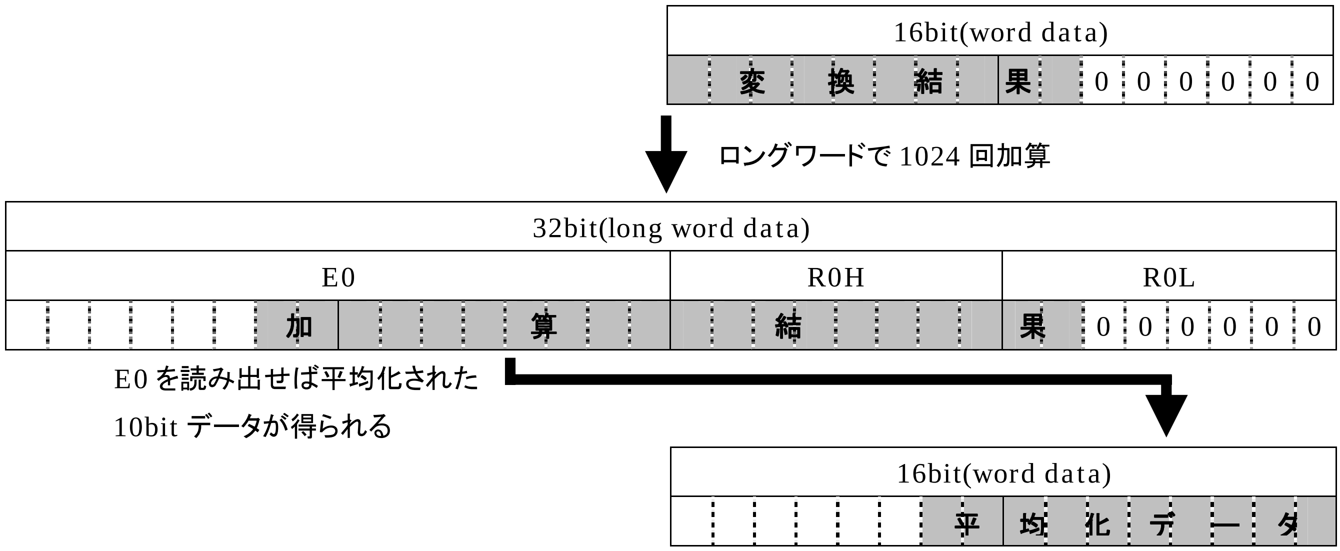
＜リスト 3-1　AD.src （1/2）＞

・平均化処理

入力されている信号源は必ずしも安定しているとは限りませんし、絶対的に安定している信号などありません。その為 A/D 変換結果もまた安定していません。そこで何回か連続して変換を行いその平均値を採用します。H8 には符号付除算命令“DIVXS”や符号無し除算命令“DIVXU”が用意されているのでここで作るプログラムでは平均化処理に除算命令を使用しています。

除算命令以外で平均化処理を行う方法もあるので紹介しておきます。これは 2 の n 乗回の平均回数時に有効です。例えば 1024 回加算してその平均値を求めるとします。2 進数では 1 ビット左にシフトすると 2 倍、右にシフトすると 1/2 倍となりなります。A/D 変換結果は 10bit ですがそのまま読み出すと下位 6bit は 0 の 16bit データです。16bit データを 1024 回加算するとその結果は 26bit のデータとなります。加算結果を 1024 で割る、つまり右に 10bit シフトすれば平均値を求められるわけですが、シフトしなくても加算結果の上位 10bit をそのまま読めば平均化したのと同じ事になります。つまり、ロングワードで加算を行い加算結果の上位ワードを読み出せば平均化された値が得られことになります。この方法は処理スピードが要求される次節で採用することにします。シフトで平均化する方法は処理が短く済む場合もありますが、逆に加算回数を任意に変えたい時には融通が利きません。H8 シリーズには除算命令が用意されているので、まずは割算でやってみることにしましょう。

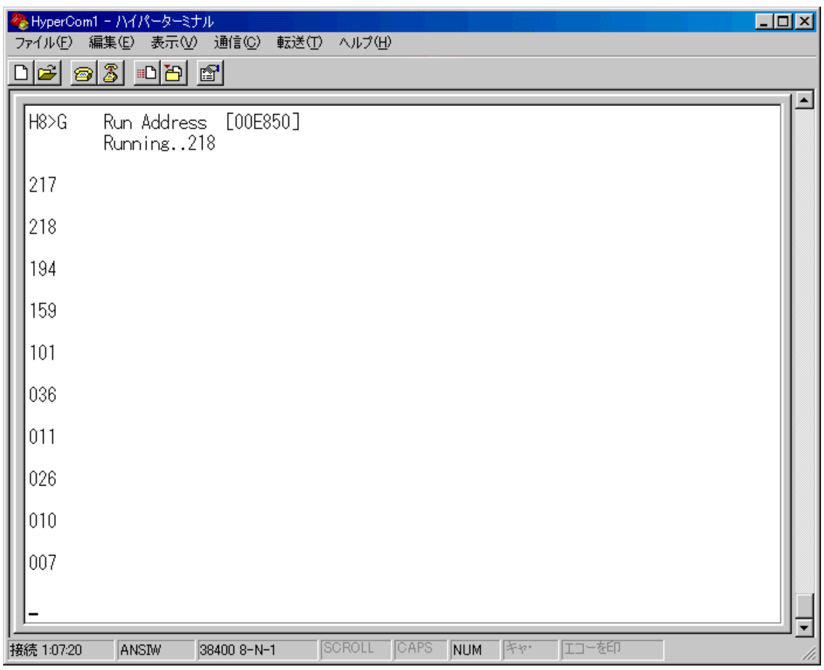
例）1024 回加算してその結果を汎用レジスタ ER0 にセットした場合



◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇◆◇

それらを踏まえて作成したプログラムが＜AD_send＞です。1 変換時間は CKS=0 なので 134 ステート、つまり 6.7 μsec ですがこれは最高速での変換時間です。実際の変換時間は約 8.3μsec でした。平均化時間を送信間隔の 1 秒間取りたいところですが、加算回数＝1sec÷8.3μsec＝120481 回。除算命令の除数は 16bit までですのでこれではオーバーしてしまいます。そこで加算回数は H’FFFF つまり 65535 回としました。平均化時間は実測で約 670msec です。実測するにあたりポート 10 にパルスを出力しています。変換タイミング及び送信間隔はタイマ V を元に作っています。送信部は巻末の付録に説明を記しますので今はそのまま使用して下さい。

次頁に＜AD_send＞のリストを示します。



```
;---------------------------------------------------------
;
;   FILE        :AD_send.src
;   DATE        :Thu, Jul 17, 2003
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1
;                 programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;---------------------------------------------------------
; ■ 概要
;   約 1sec 毎に A/D 変換を行い、その結果を 232C へ送信します。
; A/D 変換は AVE_CNST で定義した回数加算し平均化します。しかし
; 内蔵 A/D 10bit をそのまま有効にしてしまうと微細な電圧の変化
; にも反応してしまうので上位 8bit を有効数字として表示します。
; 232C の送信は上桁から順に送信し、最後に改行ｺｰﾄﾞ(CR)を送信し
; ます。
;   A/D 変換時間は 134 ｽﾃｰﾄを選択しているので 1 変換時間は、CLK
; ×134=6.7μsec ですがこれは最高速での変換時間です。1 変換時
; 間×平均化加算回数が平均値を得るまでの時間となります。実際
; には変換時間の遅れ、加算及び平均化処理が加わるので実測値は
; 計算値より大きくなります。
; （参考実測値：8.3μsec）


;-----  ｲﾝｸﾙｰﾄﾞﾌｧｲﾙ  ----------------------------------------
     .include  "io3687F_equ.inc"    ;内蔵ﾓｼﾞｭｰﾙ ｱﾄﾞﾚｽ定義

;-----  定数の定義  ----------------------------------------
;-----  SCI3  ----------------------
MHz         .equ D'20             ;X=20MHz
BAUD        .equ 38400            ;9600,19200,38400
BITR        .equ (MHz*D'1000000)/(BAUD*D'32)-1
WAIT_1B     .equ (MHz*D'1000000)/D'6/BAUD

;-----  ｱｽｷｰｺｰﾄﾞ  ----------------------------------------
BS          .equ H'08
TB          .equ H'09
LF          .equ H'0A
CL          .equ H'0C
CR          .equ H'0D

;-----  AD 変換  --------------------
AVE_CNST    .equ H'FFFF           ;平均化加算回数

;=========================================================
;          割り込みプログラム
;=========================================================
     .export    INT_TimerV

INT_TimerV:
     push.l     er0
;-----  ﾀｲﾏ V ｶｳﾝﾄ監視  ----------------------------------
TV_CHK:
     btst       #6,@TCSRV       ;ﾀｲﾏ V(1msec)ｶｳﾝﾄｱｯﾌﾟ ?
     beq        TVCHK_end       ; EQ=ｶｳﾝﾄ中

     bclr       #6,@TCSRV       ;ﾀｲﾏ V ｶｳﾝﾄｱｯﾌﾟ

     mov.w      @CNT_AD,r0      ;CNT_AD ｶｳﾝﾄ
     inc.w      #1,r0
     mov.w      r0,@CNT_AD
     cmp.w      #D'1000,r0      ;1msec * D'1000 = 1sec
     bne        TVCHK_end
     xor.w      r0,r0           ;CNT_AD ｶｳﾝﾄｱｯﾌﾟ
     mov.w      r0,@CNT_AD      ;CNT_AD ｸﾘｱ
     mov.b      #H'01,r0l       ;AD_FG ｾｯﾄ
     mov.b      r0l,@AD_FG

TVCHK_end:
     pop.l      er0
     rte

;=========================================================
;          プログラム・スタート
;=========================================================
     .export    _main
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ----------------------------------------
     bsr        INIT_PIO:16     ;PIO ｲﾆｼｬﾗｲｽﾞ
     bsr        INIT_ADC:16     ;A/D ｺﾝﾊﾞｰﾀ ｲﾆｼｬﾗｲｽﾞ
     bsr        INIT_TV:16      ;ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ / TV=ｽｷｬﾝ間隔
     bsr        INIT_SCI:16     ;SCI3 ｲﾆｼｬﾗｲｽﾞ

     andc       #H'7F,CCR       ;割り込みｲﾈｰﾌﾞﾙ

;-----  ﾒｲﾝﾙｰﾌﾟ  ----------------------------------------
Main_Loop:
     mov.b      @AD_FG,r0l      ;変換ﾌﾗｸﾞ ﾁｪｯｸ
     bne        MLoop_10        ; AD_FG=1 なら MLoop_10
     bra        Main_Loop

MLoop_10:                       ;AD_FG = 1
     xor.b      r0l,r0l         ;変換ﾌﾗｸﾞ ｸﾘｱ
     mov.b      r0l,@AD_FG
     bsr        ADC:16          ;A/D 変換
     bsr        TxD:16          ;送信
     bra        Main_Loop


;=========================================================
;          サブルーチン
;=========================================================
;---------------------------------------------------------
;          PIO ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_PIO:                       ;PIO ｲﾆｼｬﾗｲｽﾞ
     mov.b      #B'11111111,r0l ; P10-17 : 出力
     mov.b      r0l,@PCR1
     xor.b      r0l,r0l
     mov.b      r0l,@PDR1
     rts



;---------------------------------------------------------
;          A/D ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_ADC:                       ;A/D ｲﾆｼｬﾗｲｽﾞ
     mov.b      #B'00000000,r0l ; 単一ﾓｰﾄﾞ ch0 134 ｽﾃｰﾄ変換
     mov.b      r0l,@ADCSR
     rts



;---------------------------------------------------------
;          ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_TV:                        ;ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
     mov.b      #B'01001011,r0l ; ｲﾝﾀﾗﾌﾟﾄ CP-A ｲﾈｰﾌﾞﾙ
     mov.b      r0l,@TCRV0      ; ｺﾝﾍﾟｱﾏｯﾁ A ｸﾘｱ
     mov.b      #B'11100011,r0l ; 内部 CLK / 128 = 6.4usec
     mov.b      r0l,@TCRV1
     mov.b      #D'157,r0l      ;ﾀｲﾑｺﾝｽﾀﾝﾄﾚｼﾞｽﾀ A
     mov.b      r0l,@TCORA      ; = 157 * 6.4usec = 1msec
     rts
```

＜リスト 3-2 AD_send.src (1/2)＞

```
;---------------------------------------------------------
;          SCI3 ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_SCI:
     bset      #1,@PMR1
     xor.b     r0l,r0l
     mov.b     r0l,@SCR3
     mov.b     r0l,@SMR
     mov.b     #BITR,r0l
     mov.b     r0l,@BRR
     mov.w     #WAIT_1B,r0
INITSCI_00:
     dec.w     #1,r0
     bne       INITSCI_00
     mov.b     #H'30,r0l
     mov.b     r0l,@SCR3
     rts

;---------------------------------------------------------
;          A/D 変換
;---------------------------------------------------------
ADC:
;-----  計測用ﾃﾞｰﾀｾｯﾄ  -------------
     xor.w     r6,r6          ;R0L = Low
     mov.b     #H'01,r6h      ;R0H = High

     xor.l     er0,er0        ;AD_BUF ｸﾘｱ
     mov.l     er0,@AD_BUF
     mov.w     #AVE_CNST,r2   ;R2 = 加算回数
     mov.w     r2,e2          ;E2 =    〃
ADC_00:
;-----  時間計測用ﾎﾟｰﾄ出力  --------
     mov.b     r6h,@PDR1      ;P10=High

     bset      #5,@ADCSR      ;変換開始 ADST=1
ADC_02:
     btst      #7,@ADCSR      ;変換終了 ?
     beq       ADC_02
     bclr      #7,@ADCSR      ;変換終了

;-----  時間計測用ﾎﾟｰﾄ出力  --------;1 変換時間
;    mov.b     r6l,@PDR1      ;P10=Low

     mov.w     @ADDRA,r0      ;A/D 値ﾘｰﾄﾞ
     xor.w     e0,e0
     mov.l     @AD_BUF,er1
     add.l     er0,er1        ;平均化の為の加算
     mov.l     er1,@AD_BUF

     dec.w     #1,r2          ;加算終了 ?
     bne       ADC_00

;-----  時間計測用ﾎﾟｰﾄ出力  --------;平均化時間
     mov.b     r6l,@PDR1      ;P10=Low

;-----  平均化  --------------------
     mov.l     @AD_BUF,er0
     shlr.l    er0            ;ﾃﾞｰﾀ右詰め
     shlr.l    er0
     shlr.l    er0
     shlr.l    er0
     shlr.l    er0
     shlr.l    er0            ;ER0=加算 A/D ﾃﾞｰﾀ
     divxu.w   e2,er0         ;平均化 ER0/E2=R0 余り=E0

     shlr.w    r0             ;上位 8bit のみ有効とする
     shlr.w    r0
```

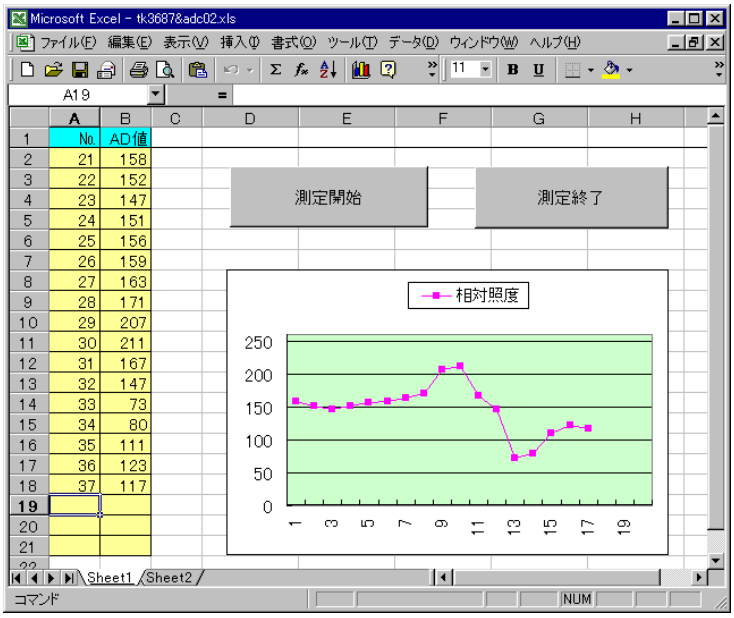
```
;-----  BCD 変換  -------------------;R0 = 平均化 A/D 値(8bit)
     mov.b     #D'100,r1l
     divxu.b   r1l,r0         ;A/D 値 % 100
     mov.b     r0l,@AD_data3  ;100 位 ｾｯﾄ

     mov.b     r0h,r0l        ;余りを R0L に移動
     xor.b     r0h,r0h        ; R0 = 余り
     mov.b     #D'10,r1l
     divxu.b   r1l,r0         ;余り % 10
     mov.b     r0l,@AD_data2  ;10 位 ｾｯﾄ

     mov.b     r0h,@AD_data1  ;余り = 1 位 ｾｯﾄ

     rts

;---------------------------------------------------------
;          A/D 送信
;---------------------------------------------------------
TxD:
     mov.b     @AD_data3,r0l
     bsr       HEX2ASCII:16
     bsr       SEND_SCI:16    ;100 位送信
     mov.b     @AD_data2,r0l
     bsr       HEX2ASCII:16
     bsr       SEND_SCI:16    ;10 位送信
     mov.b     @AD_data1,r0l
     bsr       HEX2ASCII:16
     bsr       SEND_SCI:16    ;1 位送信
     mov.b     #CR,r0l
     bsr       SEND_SCI:16    ;改行送信
TxD_end:
     rts

;---------------------------------------------------------
;          SEND_SCI : 送信
;---------------------------------------------------------
SEND_SCI:
     btst      #7,@SSR
     beq       SEND_SCI
     mov.b     r0l,@TDR
     rts

;---------------------------------------------------------
;          HEX -> ASCII 変換
;---------------------------------------------------------
HEX2ASCII:                    ;R0L=ﾃﾞｰﾀ(bit4-7=0)
     add.b     #H'30,r0l      ;数字へｱｽｷｰ変換
     cmp.b     #H'3A,r0l
     bcs       HEX2ASCII_00   ;A-F?
     add.b     #H'07,r0l      ;A-F へｱｽｷｰ変換
HEX2ASCII_00:
     rts

;=========================================================
;          データ・エリア
;=========================================================
     .section  D,data,locate=H'FD00
;-----  A/D ｺﾝﾊﾞｰﾀ  ----------------------------------------
AD_FG:        .res.b    1    ;変換ﾌﾗｸﾞ
     .align    2
CNT_AD:       .res.w    1    ;A/D 変換ﾀｲﾐﾝｸﾞ ｶｳﾝﾀ
AD_BUF:       .res.l    1    ;A/D 加算ﾊﾞｯﾌｧ
AD_data:                     ;BCD 化 A/D 値
AD_data1:     .res.b    1
AD_data2:     .res.b    1
AD_data3:     .res.b    1
     .align    2

     .end
```

＜リスト 3-2 AD_send.src (2/2)＞

■VBA を用いた Excel への変換結果取り込み

CD-R 内のフォルダ Excel&VBA に収録されている Excel ファイル“tk3687&adc01.xls”は、VBA を用いて変換結果を Excel ワークシート内に取り込むサンプルプログラムです。このマクロを応用し Excel にデータを取り込むことで変換結果を収集・グラフ化することが出来ますので、色々と応用が拡がります。

マクロの詳細については VisualBasicEditor を開き、コードまたはオブジェクトウィンドウを参照して下さい。

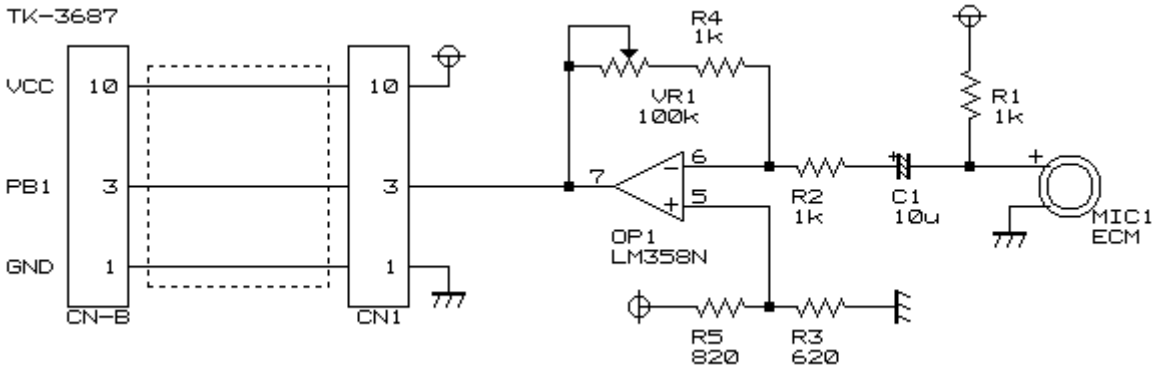


4-1 簡易レベルメータのプログラム

この章では音というアナログ量をコンデンサマイクで電圧に変換し、入力された音の大きさによって TK-3687 上の LED を点灯させる簡易レベルメータをプログラムします。3 章で扱った光センサのような直流入力とは違い、音という交流信号を A/D 変換します。変換方法自体は同じですが、音は交流ですのである値を境に＋と－に信号は振れています。そこで－側に振れた信号はプログラムによって＋側へと変換し、短区間の平均値を取って瞬時々の音量レベルとします。ここで、短区間の平均値を求めるということはローパスフィルタを通すことを意味します。

■ハード

図 4-1 は音声入力部を抜き出した回路図です。コンデンサマイクより入力された信号はオペアンプによって 1～100 倍に増幅され TK-3687 の PB1 に入力されます。オペアンプのオフセットは約 2.2V でこれは音声無入力時の実測値によって決定しました。



＜図 4-1 音声入力部回路図＞

■ソフト

使用するレジスタは 3 章で説明したレジスタと同じ ADCSR,ADDRA～D です。但し、今回は変換時間を短くしたいので ADCSR の CKS ビットを 1 にします。尚、入力に PB1 つまり AN1 を使用しているので読み出す A/D データレジスタは ADDRB です。A/D コントロール／ステータスレジスタは次のようになります。

・ADCSR （A/D コントロール／ステータスレジスタ）

7	6	5	4	3	2	1	0
ADF	**ADIE**	**ADST**	**SCAN**	**CKS**	**CH2**	**CH1**	**CH0**
0	**0**	**0**	**0**	**1**	**0**	**0**	**1**

・ADDRB（A/D データレジスタ B）

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
変換結果(10bit)										**0**	**0**	**0**	**0**	**0**	**0**

・平均化処理

今回は 10bit 全て使用します。平均化加算回数は 16 回とし、(1 変換時間×16 回)時間の平均値を取ります。1 変換時間は 70 ステートですので平均化時間は CLK×70 ステート×16 回＝56μsec となりそうですが、これはあくまで理想値です。実際には変換時間の遅れや平均化の処理が加わり 56μsec 以上の時間がかかり、実測値で約 110μsec かかっていました。加算は前回同様ロングワードで行い符号無し除算命令“DIVXU”を用いて平均化します。尚、この平均化はノイズ除去の為の平均化で、後述の整流の為の平均化とは別物であることに注意して下さい。

・絶対値化

前記しましたように音は交流信号なのに対してレベルメータは直流変化なので得られた A/D 値を絶対値化する必要があります。無音時の A/D 値を基準に、この値より低い値は－方向(谷波形)とし算術によって＋方向(山波形)に変換します。まず得られた A/D 値と無音時の A/D 値“Silent_Lv”を減算します。A/D 値が“Silent_Lv”よりより大きければそのまま計算結果を採用し、逆に小さければ 2 の補数を求め＋側に変換します。変換部分のプログラムを以下に示します。

```
;-------------------------------------------------------------
;          絶対値化
;-------------------------------------------------------------
abs16:
     mov.w     @AD_data,r0          ;R0=平均化 A/D 値
     mov.w     #Silent_Lv,e0        ;E0=無音時ﾚﾍﾞﾙ

     sub.w     e0,r0                ;山波形?
     bcc       abs16_00             ; 山波形ならそのまま
     not.w     r0                   ;谷波形
     inc.w     #1,r0                ; 波形反転
abs16_00:
     mov.w     r0,@absAD_data
     rts
```

・表示

LED は 8 個並んでいるので、平均化された A/D 値を 8 つのデータと比較して表示データを作成、出力します。

◆◇

今までの要素を踏まえた上で作成したプログラムが＜**AD_Lvmeter01**＞です。レベルインジゲータには P5 の LED を使用することにします。次頁にリストを示します。

```
;---------------------------------------------------------
;
;   FILE        :AD_Lvmeter01.src
;   DATE        :Thu, Feb 12, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1)
;                 programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;---------------------------------------------------------
; ■ 概要
;   AN1 に入力された音声信号をレベル判定をし、LED を点灯させます。 入力信号の電圧範囲は約 2.2V を基準に±1.5V です。基準値
; は A/D 値で H'01C2 ですのでそれ以下の谷波形は反転してからレベル判定を行います。ピーク値は H'14E なのでその値近くで最上位
; bit の LED が点灯するように判定値を設定します。
; 判定値は“CP_TBL”で定義されています。ﾉｲｽﾞ除去の為の平均化回数は 16 回で変換時間は (1/CLK)x(70max)x16 = 56μsec(max) ですが、実際には 1 変換時間の遅れ・平均化処理が加わるので約 110μsec(実測値)です。
;■ＶＲの調整
; 入力される最大音量で LED が全点灯するように VR を調節します。あまりにも LED の点灯が少ない場合は VR を上げ、逆に頻繁に全点灯するようであれば VR を下げてください。


;----- ｲﾝｸﾙｰﾄﾞﾌｧｲﾙ ----------------------------------------
     .include  "io3687F_equ.inc"    ;H8/3687 I/O 定義ﾌｧｲﾙ

;----- 定数の定義 ------------------------------------------
AVE_CNST        .equ  D'16        ;加算回数
Silent_Lv       .equ  H'01C2      ;無音時ﾚﾍﾞﾙ(初期値)

;=========================================================
;          プログラム・スタート
;=========================================================
     .export   _main
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------------------------------
     bsr       INIT_ADC:16     ;A/D ｺﾝﾊﾞｰﾀ ｲﾆｼｬﾗｲｽﾞ
     bsr       INIT_PIO:16     ;PIO ｲﾆｼｬﾗｲｽﾞ

;-----  ﾒｲﾝﾙｰﾌﾟ  ------------------------------------------
Main_Loop:
     bsr       ADC:16          ;A/D 変換と平均化
     bsr       abs16:16        ;絶対値化
     bsr       DISP5:16        ;ﾎﾟｰﾄ 5 へ表示
     bra       Main_Loop


;=========================================================
;          サブルーチン
;=========================================================
;---------------------------------------------------------
;          A/D ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_ADC:                        ;A/D ｲﾆｼｬﾗｲｽﾞ
     mov.b     #B'00001001,r0l ; 単一ﾓｰﾄﾞ AN1 7Ch ｽﾃｰﾄ
     mov.b     r0l,@ADCSR
     rts

;---------------------------------------------------------
;          PIO ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_PIO:                        ;PIO ｲﾆｼｬﾗｲｽﾞ
     mov.b     #H'FF,r0l       ;P5 = 表示
     mov.b     r0l,@PCR5
     mov.b     #B'11111111,r0l ; P10-17 : 出力
     mov.b     r0l,@PCR1
     xor.b     r0l,r0l
     mov.b r0l,@PDR1

     rts

;---------------------------------------------------------
;          A/D 変換
;---------------------------------------------------------
ADC:
;-----  計測用ﾃﾞｰﾀｾｯﾄ  -------------
     xor.w     r6,r6           ;R0L = Low
     mov.b     #H'01,r6h       ;R0H = High

     xor.l     er0,er0         ;AD_BUF ｸﾘｱ
     mov.l     er0,@AD_BUF
     mov.w     #AVE_CNST,r2    ;R2 = 加算回数
     mov.w     r2,e2
ADC_00:
;-----  時間計測用ﾎﾟｰﾄ出力  --------
     mov.b     r6h,@PDR1       ;P10=High

     bset      #5,@ADCSR       ;変換開始
ADC_02:
     btst      #7,@ADCSR       ;変換終了 ?
     beq       ADC_02
     bclr      #7,@ADCSR       ;変換終了

     mov.w     @ADDRB,r0       ;A/D 値ﾘｰﾄﾞ
     xor.w     e0,e0
     mov.l     @AD_BUF,er1
     add.l     er0,er1         ;平均化の為の加算
     mov.l     er1,@AD_BUF

     dec.w     #1,r2           ;加算終了 ?
     bne       ADC_00

;-----  時間計測用ﾎﾟｰﾄ出力  --------;平均化時間
     mov.b     r6l,@PDR1       ;P10=Low

;-----  平均化  --------------------
     mov.l     @AD_BUF,er0
     shlr.l    er0             ;ﾃﾞｰﾀ右詰め
     shlr.l    er0
     shlr.l    er0
     shlr.l    er0
     shlr.l    er0
     shlr.l    er0             ;ER0=加算 A/D ﾃﾞｰﾀ
     divxu.w   e2,er0          ;平均化 ER0/E2=R0 余り=E0
     and.w     #H'03FF,r0      ;10bit 有効

     mov.w     r0,@AD_data     ;R0=平均化 A/D 値
     rts

;---------------------------------------------------------
;          絶対値化
;---------------------------------------------------------
abs16:
     mov.w     @AD_data,r0     ;R0=平均化 A/D 値
     mov.w     #Silent_Lv,e0   ;E0=無音時ﾚﾍﾞﾙ

     sub.w     e0,r0           ;山波形?
     bcc       abs16_00        ; 山波形ならそのまま
     not.w     r0              ;谷波形
     inc.w     #1,r0           ; 波形反転
abs16_00:
     mov.w     r0,@absAD_data
     rts
```

＜リスト 4-1 AD_Lvmeter01.src (1/2)＞

```
;---------------------------------------------------------
;          LED へ表示
;---------------------------------------------------------
DISP5:
     mov.w     @absAD_data,r0  ;R0  = A/D 値
     mov.l     #CP_TBL,er3     ;ER3 = 比較ﾃｰﾌﾞﾙｱﾄﾞﾚｽ
     xor.b     r1l,r1l         ;R1L = 表示ﾃﾞｰﾀ
DISP5_00:
     mov.w     @er3,e0
     cmp.w     e0,r0           ;比較
     bcs       DISP5_02
     shll.b    r1l             ;表示ﾃﾞｰﾀｼﾌﾄ
     bset      #0,r1l          ;点灯 bit ｾｯﾄ
     inc.l     #2,er3          ;Address+2:次の比較値へ
     bra       DISP5_00

DISP5_02:
     mov.br1l,@PDR5            ;表示
     rts
```

```
CP_TBL:
     .data.w   D'3,D'5,D'10,D'21
     .data.w   D'41,D'83,D'165,D'330
     .data.w   H'FFFF


;=========================================================
;          データ・エリア
;=========================================================
     .section  D,data,locate=H'FD00

AD_BUF:         .res.l    1     ;A/D 加算 ﾊﾞｯﾌｧ
AD_data:        .res.w    1     ;平均化 A/D ﾃﾞｰﾀ
absAD_data:     .res.w    1     ;絶対値化 A/D ﾃﾞｰﾀ


;/////////////////////////////////////////////////////////
          .end
```

<リスト 4-1 AD_Lvmeter01.src (2/2)>

4-2 キャリブレーション

4-1 のプログラムでは無音時、つまり零点の値を定数 Silent_Lv として定義していました。Silent_Lv の値はコンデンサマイクを接続しない時の A/D 値から定数を決定しましたが、部品のバラつきや使用する環境によってこの値は必ずしも正しいとは限りません。又、VR でアンプのゲインを変更することによっても変わります。そこで A/D 基板上のスイッチを用いて零点のキャリブレーションを行わせましょう。

キャリブレーションは整流する前の A/D 値を一定時間サンプリングし平均化することで零点を求める事が出来ます。従って、A/D 変換を 2^{13}=8192 回行いその平均値を求め、それを SilentLv（アナログの仮想 GND）とします。キャリブレーションに要する時間は（1 変換時間 110μsec＋加算処理等約 3.9μsec）×加算回数 8192 回=約 933msec 程でその間は音を立ててはなりません。キャリブレーションのプログラムを以下に示します。

```
;---------------------------------------------------------------------
;          キャリブレーション
;---------------------------------------------------------------------
Calibration:
     mov.w     #D'8192,r3          ;加算回数 : D'8192=2 の 13 乗
     xor.l     er4,er4             ;ER4=加算ﾊﾞｯﾌｧ
Calibration_00:
     bsr       ADC:16              ;ﾉｲｽﾞ除去された 1 変換ﾃﾞｰﾀ -> R0
     xor.w     e0,e0
     add.l     er0,er4             ;加算
     dec.w     #1,r3
     bne       Calibration_00
     shll.l    er4                 ;平均化
     shll.l    er4
     shll.l    er4
     mov.w     e4,@SilentLv_BUF
     rts
```

・スイッチ読み込み

A/D 基板上に実装されているスイッチは機械式接点ですので押した時に接点がバウンドして何回も ON／OFF を繰り返すチャタリングという現象を起こします。チャタリングは人間の感覚では問題にならないのですがマイコンの

実行速度ですとスイッチが何度も押されたように読み取ってしまうので対策が必要です。そこで、スイッチが押されたら少し時間を置いてから再度読み込み、最初に読み取った状態と同じならスイッチが押されたと判断するダブルリード処理を行います。このプログラムでは最初にスイッチが押されたらその値をバッファ Sw_1st に蓄えタイマ V で 5msec 間隔を置きダブルリードを行います。2 度目に読み込んだ値と Sw_1st とが同じであればスイッチが押されたと判断しスイッチ番号 Sw_No をセットします。尚、スイッチが押された時のみ反応するようにエッジ検出処理も入れました。このエッジ検出はダブルリードでスイッチが押されているのを確定した後、Sw_No と押されたスイッチとを比較しと同じであれば押し続けと判断します。尚、キャリブレーションは 3 つスイッチの内いずれかが押されたら行うようにしました。

キャリブレーション機能を追加した**＜AD_Lvmeter02＞**のリストを以下に示します。尚、今回の平均化は処理スピードを早くする為に割算ではなくシフト命令だけで行うことにします。

```
;---------------------------------------------------------
;
;   FILE        :AD_Lvmeter02.src
;   DATE        :Thu, Feb 12, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1)
;                 programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;---------------------------------------------------------
; ■ 概要
;   AN1 に入力された音声信号をレベル判定をし、LED を点灯させます。 入力信号の電圧範囲は約 2.2V を基準に±1.5V です。基準値は A/D 値で H'01C2 ですのでそれ以下の谷波形は反転してからレベル判定を行います。ピーク値は H'14E なのでその値近くで最上位 bit の LED が点灯するように判定値を設定します。
; 判定値は“CP_TBL”で定義されています。ﾉｲｽﾞ除去の為の平均化回数は 16 回で変換時間は (1/CLK)x(70max)x16 = 56μsec(max) ですが、実際には 1 変換時間の遅れ・平均化処理が加わるので約 110μsec(実測値)です。
;■キャリブレーション
;   A/D 基板上のスイッチいずれかを押すことで、絶対値化を行う際の基準値を更正します。デフォルトで H'01C2 がセットされます。
;■ＶＲの調整
; 入力される最大音量で LED が全点灯するように VR を調節します。あまりにも LED の点灯が少ない場合は VR を上げ、逆に頻繁に全点灯するようであれば VR を下げてください。


;----- ｲﾝｸﾙｰﾄﾞﾌｧｲﾙ ------------------------------------------
    .include  "io3687F_equ.inc"    ;H8/3687 I/O 定義ﾌｧｲﾙ

;----- 定数の定義 ------------------------------------------
AVE_CNST        .equ  D'16       ;加算回数
Silent_Lv       .equ  H'01C2     ;無音時ﾚﾍﾞﾙ(初期値)


;=========================================================
;          割り込みプログラム
;=========================================================
    .export   INT_TimerV

INT_TimerV:
    push.l    er0
;-----  ﾀｲﾏ V ｶｳﾝﾄ監視  ----------------------------------
TVCHK:
    bclr      #6,@TCSRV       ;ﾀｲﾏ V ｶｳﾝﾄｱｯﾌﾟ
TVCHK_00:
    mov.b     @Sw_STA,r0l     ;Sw_STA ﾁｪｯｸ
    cmp.b     #1,r0l          ; Sw_STA=1 ならｶｳﾝﾄ
    beq       TVCHK_Sw
TVCHK_02:
    pop.l     er0
    rte

;-----  Sw ｳｪｲﾄｶｳﾝﾄ  ---------------
TVCHK_Sw:                     ;Sw_STA=1 : ﾀﾞﾌﾞﾙﾘｰﾄﾞ ｳｪｲﾄ中
    mov.b     @SwWaitCNT,r0l  ;SwWaitCNT ｶｳﾝﾄ
    inc.b     r0l             ; ｳｪｲﾄ時間=5msec
    cmp.b     #5,r0l          ;ｶｳﾝﾄｱｯﾌﾟ？
    bcs       TVCHK_Sw00      ; Cy=0 : ｶｳﾝﾄｱｯﾌﾟ
    mov.b     #2,r0l          ;Sw_STA 次のｽﾃｰｼﾞへ
    mov.b     r0l,@Sw_STA     ; Sw_STA=2
    xor.b     r0l,r0l         ;ｶｳﾝﾄ ｸﾘｱ
TVCHK_Sw00:
    mov.b     r0l,@SwWaitCNT  ;ｳｪｲﾄｶｳﾝﾀ ｸﾘｱ
    bra       TVCHK_02

;=========================================================
;          プログラム・スタート
;=========================================================
    .export   _main
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ----------------------------------------
    bsr       INIT_ADC:16     ;A/D ｺﾝﾊﾞｰﾀ ｲﾆｼｬﾗｲｽﾞ
    bsr       INIT_TV:16      ;ﾀｲﾏ V INIT
    bsr       INIT_PIO:16     ;PIO ｲﾆｼｬﾗｲｽﾞ

    xor.l     er0,er0
    mov.l     er0,@AD_BUF     ;A/D 加算ﾊﾞｯﾌｧ ｸﾘｱ
    mov.w     r0,@absAD_data  ;絶対値化 A/D ﾊﾞｯﾌｧ ｸﾘｱ
    mov.b     r0l,@Sw_STA     ;Sw ｽﾃｰﾀｽ ｸﾘｱ
    mov.b     r0l,@SwWaitCNT  ;Sw ｳｪｲﾄｶｳﾝﾀ ｸﾘｱ
    mov.b     r0l,@Sw_FG      ;Sw ﾌﾗｸﾞ ｸﾘｱ

    mov.w     #Silent_Lv,r0   ;無音値の初期化
    mov.w     r0,@SilentLv_BUF

    andc      #H'7F,CCR       ;割り込みｲﾈｰﾌﾞﾙ

;-----  ﾒｲﾝﾙｰﾌﾟ  ----------------------------------------
Main_Loop:
    bsr       ADC:16          ;A/D 変換と平均化
    bsr       abs16:16        ;絶対値化
    bsr       SwRead:16       ;Sw ﾘｰﾄﾞ
```

＜リスト 4-2 AD_Lvmeter02.src (1/3)＞

```
     mov.b     @Sw_FG,r0l      ;Sw_FG ﾁｪｯｸ
     bne       MLoop_10        ; Sw_FG=1 : Sw 押された
     bsr       DISP5:16        ;ﾎﾟｰﾄ 5 へ表示
     bra       Main_Loop

;-----  ｽｲｯﾁ入力有り  --------------
MLoop_10:
     bsr       Calibration:16  ;ｷｬﾘﾌﾞﾚｰｼｮﾝ
     bra       Main_Loop

;=========================================================
;          サブルーチン
;=========================================================
;---------------------------------------------------------
;          A/D ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_ADC:                      ;A/D ｲﾆｼｬﾗｲｽﾞ
     mov.b     #B'00001001,r0l ; 単一ﾓｰﾄﾞ AN1 70 ｽﾃｰﾄ
     mov.b     r0l,@ADCSR
     rts

;---------------------------------------------------------
;          ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_TV:                       ;ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
     mov.b     #B'01001011,r0l ; ｲﾝﾀﾗﾌﾟﾄ CP-A ｲﾈｰﾌﾞﾙ
     mov.b     r0l,@TCRV0      ; ｺﾝﾍﾟｱﾏｯﾁ A ｸﾘｱ
     mov.b     #B'11100011,r0l ; 内部 CLK / 128 = 6.4usec
     mov.b     r0l,@TCRV1
     mov.b     #D'157,r0l      ;ﾀｲﾑｺﾝｽﾀﾝﾄﾚｼﾞｽﾀ A
     mov.b     r0l,@TCORA      ; = 157 * 6.4usec = 1msec
     rts

;---------------------------------------------------------
;          PIO ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_PIO:                      ;PIO ｲﾆｼｬﾗｲｽﾞ
     mov.b     #H'FF,r0l       ;P5 = 表示
     mov.b     r0l,@PCR5

     mov.b     #B'11111111,r0l ; P10-17 : 出力
     mov.b     r0l,@PCR1
     xor.b     r0l,r0l
     mov.b     r0l,@PDR1

     rts

;---------------------------------------------------------
;          A/D 変換
;---------------------------------------------------------
ADC:
     orc       #H'80,CCR       ;割り込みﾃﾞｨｲﾈｰﾌﾞﾙ

     xor.l     er0,er0         ;AD_BUF ｸﾘｱ
     mov.l     er0,@AD_BUF
     mov.w     #AVE_CNST,r2    ;R2 = 加算回数
     mov.w     r2,e2
ADC_00:
     bset      #5,@ADCSR       ;変換開始
ADC_02:
     btst      #7,@ADCSR       ;変換終了 ?
     beq       ADC_02
     bclr      #7,@ADCSR       ;変換終了

     mov.w     @ADDRB,r0       ;A/D 値ﾘｰﾄﾞ
     xor.w     e0,e0
     mov.l     @AD_BUF,er1
     add.l     er0,er1         ;平均化の為の加算
     mov.l     er1,@AD_BUF

     dec.w     #1,r2           ;加算終了 ?
     bne       ADC_00

     andc      #H'7F,CCR       ;割り込みｲﾈｰﾌﾞﾙ

;-----  平均化  --------------------
     mov.l     @AD_BUF,er0
;-----  ｼﾌﾄによる平均化  -----------;平均化回数=16 回
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     mov.w     e0,r0

     mov.w     r0,@AD_data     ;R0=平均化 A/D 値
     rts

;---------------------------------------------------------
;          絶対値化
;---------------------------------------------------------
abs16:
     mov.w     @AD_data,r0     ;R0=平均化 A/D 値
     mov.w     @SilentLv_BUF,e0     ;E0=無音時ﾚﾍﾞﾙ

     sub.w     e0,r0           ;山波形?
     bcc       abs16_00        ; 山波形ならそのまま
     not.w     r0              ;谷波形
     inc.w     #1,r0           ; 波形反転
abs16_00:
     mov.w     r0,@absAD_data
     rts

;---------------------------------------------------------
;          ｽｲｯﾁﾘｰﾄﾞ
;---------------------------------------------------------
SwRead:
     mov.b     @Sw_STA,r0l
     cmp.b     #0,r0l
     beq       SwRead_STA0
     cmp.b     #1,r0l
     beq       SwRead_STA1
     cmp.b     #2,r0l
     beq       SwRead_STA2

     xor.b     r0l,r0l
     mov.b     r0l,@Sw_STA
     mov.b     r0l,@Sw_FG
     rts

;----- 1st ﾘｰﾄﾞ --------------------
SwRead_STA0:                   ;ｽﾃｰﾀｽ=0
     mov.b     @PDRB,r0l       ;Sw ﾁｪｯｸ PB ﾘｰﾄﾞ
     not.b     r0l             ;反転
     and.b     #B'01110000,r0l ;Sw = bit6-4
     beq       SwRead_off      ;EQ = 押されていない
     mov.b     r0l,@Sw_1st     ;ﾀﾞﾌﾞﾙﾘｰﾄﾞの為 退避
     mov.b     #1,r0l          ;次のｽﾃｰﾀｽへ
     mov.b     r0l,@Sw_STA     ; Sw_STA=1
     rts

;----- ｳｪｲﾄ ------------------------
SwRead_STA1:                   ;ｽﾃｰﾀｽ=1
     rts                       ;ﾀｲﾏ V によるｶｳﾝﾄ中

;----- 2nd ﾘｰﾄﾞ --------------------
SwRead_STA2:                   ;ｽﾃｰﾀｽ=2
```

<リスト 4-2 AD_Lvmeter02.src (2/3)>

```
     xor.b     r0l,r0l          ;Sw_STA ｸﾘｱ
     mov.b     r0l,@Sw_STA

     mov.b     @PDRB,r0l        ;Sw ﾁｪｯｸ PB ﾘｰﾄﾞ
     not.b     r0l              ;反転
     and.b     #B'01110000,r0l ;Sw=bit6-4
     beq       SwRead_off       ;EQ=押されていない
     mov.b     @Sw_1st,r1l
     cmp.b     r0l,r1l          ;1st ﾘｰﾄﾞと同じ?
     bne       SwRead_off
;----- 入力確定 --------------------
     mov.b     #1,r0l
     btst      #4,r1l           ;Bit4=Sw1 : ROL=1
     bne       SwRead_on
     inc.b     r0l
     btst      #5,r1l           ;Bit5=Sw2 : ROL=2
     bne       SwRead_on
     inc.b     r0l
     btst      #6,r1l           ;Bit6=Sw3 : ROL=3
     beq       SwRead_off
;----- 入力有り --------------------
SwRead_on:
;----- ﾜﾝｼｮｯﾄ動作 ------------------
     mov.b     @Sw_No,r0h       ;押し続けは無視する
     cmp.b     r0l,r0h          ;前回と一致？
     bne       SwRead_on1
     xor.b     r0l,r0l          ; 押し続け : Sw_FG=0
     bra       SwRead_FS
;----- Sw 番号確定 ------------------
SwRead_on1:
     mov.b     r0l,@Sw_No       ;Sw_No ｾｯﾄ
     mov.b     #1,r0l           ;Sw_FG = 1
     bra       SwRead_FS

;----- 入力無し --------------------
SwRead_off:
     xor.b     r0l,r0l
     mov.b     r0l,@Sw_No       ;ｽｲｯﾁ番号ｸﾘｱ
SwRead_FS:
     mov.b     r0l,@Sw_FG
     rts

;-------------------------------------------------------
;          キャリブレーション
;-------------------------------------------------------
Calibration:
;-----  計測用ﾃﾞｰﾀｾｯﾄ  -------------
     xor.w     r6,r6            ;ROL = Low
     mov.b     #H'01,r6h        ;ROH = High

;-----  時間計測用ﾎﾟｰﾄ出力  --------
     mov.b     r6h,@PDR1        ;P10=High

     mov.w     #D'8192,r3       ;加算回数 : D'8192=2 の 13 乗
     xor.l     er4,er4          ;ER4=加算ﾊﾞｯﾌｧ
Calibration_00:
     bsr       ADC:16           ;ﾉｲｽﾞ除去された 1 変換ﾃﾞｰﾀ->R0
     xor.w     e0,e0
     add.l     er0,er4          ;加算
     dec.w     #1,r3
     bne       Calibration_00
     shll.l    er4              ;平均化
     shll.l    er4
     shll.l    er4
     mov.w     e4,@SilentLv_BUF
```

```
;-----  時間計測用ﾎﾟｰﾄ出力  --------;ｷｬﾘﾌﾞﾚｰｼｮﾝ時間
     mov.b     r6l,@PDR1        ;P10=Low

     rts

;-------------------------------------------------------
;          LED へ表示
;-------------------------------------------------------
DISP5:
     mov.w     @absAD_data,r0  ;R0  = A/D 値
     mov.l     #CP_TBL,er3      ;ER3 = 比較ﾃｰﾌﾞﾙｱﾄﾞﾚｽ
     xor.b     r1l,r1l          ;R1L = 表示ﾃﾞｰﾀ
DISP5_00:
     mov.w     @er3,e0
     cmp.w     e0,r0            ;比較
     bcs       DISP5_02
     shll.b    r1l              ;表示ﾃﾞｰﾀｼﾌﾄ
     bset      #0,r1l           ;点灯 bit ｾｯﾄ
     inc.l     #2,er3           ;Address+2:次の比較値へ
     bra       DISP5_00
DISP5_02:
     mov.b     r1l,@PDR5        ;表示
     rts

CP_TBL:
     .data.w   D'3,D'5,D'10,D'21
     .data.w   D'41,D'83,D'165,D'330
     .data.w   H'FFFF

;=======================================================
;          データ・エリア
;=======================================================
     .section  D,data,locate=H'FD00

;-----  A/D  ---------------------------------------------
AD_BUF:         .res.l    1     ;A/D 加算 ﾊﾞｯﾌｧ
AD_data:        .res.w    1     ;平均化 A/D ﾃﾞｰﾀ
absAD_data:     .res.w    1     ;絶対値化 A/D ﾃﾞｰﾀ

SilentLv_BUF:   .res.w    1     ;無音値

;-----  Sw  ----------------------------------------------
Sw_STA:         .res.b    1     ;ｽﾃｰﾀｽ
SwWaitCNT:      .res.b    1     ;ｳｪｲﾄ ｶｳﾝﾀ
Sw_1st:         .res.b    1     ;初回に押されたｽｲｯﾁ状態
Sw_FG:          .res.b    1     ;ｽｲｯﾁ ﾌﾗｸﾞ 1=押された
Sw_No:          .res.b    1     ;ｽｲｯﾁ No,
          .align    2


;=======================================================
          .end
```

<リスト 4-2 AD_Lvmeter02.src (3/3)>

5 整流値・ピーク値の送信

4 章では絶対値化した音声を LED に表示しました。この章では音声を整流し 232C で送信、ハイパーターミナルで表示してみましょう。タイマ V で約 0.1 秒の送信タイミングを作り、その間の A/D 値を整流し送信します。しかしただ変換結果を送信するだけではあまり芸がありません。そこで今までの最大値を記憶しておき、最大値が更新された時のみ送信・表示を更新するピーク値の送信も追加します。機能の切り替えにはいくつか方法がありますがここでは A/D 基板上のスイッチを用い、いずれかが押されたら表示を切り替えます。

■ソフト

・ノイズ除去の為の平均化、及び絶対値化

・スイッチの読み取り

4 章と同じです。

・零点のキャリブレーション

4-2 章ではキャリブレーションをスイッチによるマニュアルで行っていましたが、ここではイニシャライズの一つとしてプログラム実行時に自動的にキャリブレーションを行わせます。メインループに入る前にコールしていること以外は 4-2 章と同じ処理です。

・整流値

送信の間の A/D 値を平均化することで整流を行います。送信間隔を 0.1sec 程度にしたいので平均化の加算回数は、0.1sec÷（ノイズを除去した 1 データの変換時間 110μsec）=約 909 回となります。平均化処理をシフト命令で行う都合上 2^n の方がよいので加算回数を 1024 回とすると整流時間は約 112msec となり、0.1 秒を越えてしまいます。そこで加算回数は 512 回とします。

・ピーク値

ピーク値はノイズ除去した 1 データの最高値をバッファに蓄え、1 データ変換後とに比較します。表示は最高値が更新された時のみ送信・表示します。

・タイマ V

タイマ V は 1msec 毎に割り込みがかかるようにします。内部では 2 つのカウンタを持たせ、1 つはスイッチのダブルリード時に動くカウンタ、そしてもう 1 つは整流値の送信タイミングを作るカウンタです。いずれのカウンタもフラグによって動作を管理します。

・ 表示

表示の為の送信データは、まず最初に CL コード送信して画面をクリアし、次に A/D 値をアスキー変換して送信します。

◆◇

以上を踏まえて作成したのが＜AD_Peak.src＞です。スイッチによる判定が加わった分 4 章と比べて少し複雑になっていますが、動作のきっかけはフラグによってコントロールされています。フラグのセット・リセットを把握すればそれほど難しくはありません。次頁にリストを示します。

```
;---------------------------------------------------------
;
;   FILE        :AD_Peak.src
;   DATE        :Mon, Feb 09, 2004
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1)
;
;---------------------------------------------------------
; ■ 概要
;   AN1 に入力された音声信号を整流し、整流値又はﾋﾟｰｸ値を 232C
; で送信します。表示の切り替えは基板上のｽｲｯﾁで行い、いずれかが
; 押されると切り替えます。ﾉｲｽﾞ除去の為の平均化回数は 16 回で変
; 換時間は (1/CLK)x(70max)x16 = 56μsec(max) ですが、実際には
; 1 変換時間の遅れ・平均化処理が加わるので約 110μsec(実測値)
; です。
; ■ ﾋﾟｰｸ値の送信
;   Peak_FG=1 の時、変換結果と最大値とを比較、最大値が更新され
; たら 232C で送信します。表示は改行ではなく一旦クリアしてから
; 最大値を表示します。
; ■ タイマＶ
;   タイマＶは送信間隔、及びスイッチのダブルリード時間を管理し
; ています。
; ■ VR の調整
;   入力する音に対してあまり値が変わらないようであれば VR を上
; げて下さい。逆にﾋﾟｰｸ値(表示上で約 330)にすぐ達してしまう場合
; は VR を下げて下さい。

;----- ｲﾝｸﾙｰﾄﾞﾌｧｲﾙ ------------------------------------------
    .include  "io3687F_equ.inc"    ;H8/3687 I/O 定義ﾌｧｲﾙ

;----- 定数の定義 --------------------------------------------
;-----  SCI3  ----------------------
MHz       .equ D'20       ;X=20MHz
BAUD      .equ 38400      ;9600,19200,38400
BITR      .equ (MHz*D'1000000)/(BAUD*D'32)-1
WAIT_1B   .equ (MHz*D'1000000)/D'6/BAUD

;-----  AD 変換  ---------------------
AVE_CNST  .equ D'16       ;加算回数
REC_CNST  .equ D'512      ;整流値算出回数
Silent_Lv .equ H'01C2     ;無音時ﾚﾍﾞﾙ

;-----  送信  ----------------------
TxTiming  .equ D'100      ;送信間隔 : 1msec x 100 = 100msec

;-----  ｱｽｷｰｺｰﾄﾞ  ---------------------------------------
BS        .equ H'08
TB        .equ H'09
LF        .equ H'0A
CL        .equ H'0C
CR        .equ H'0D



;=========================================================
;          割り込みプログラム
;=========================================================
    .export   INT_TimerV

INT_TimerV:
    push.l    er0
;-----  ﾀｲﾏV ｶｳﾝﾄ監視  ------------------------------------
TVCHK:
    bclr      #6,@TCSRV       ;ﾀｲﾏ V ｶｳﾝﾄｱｯﾌﾟ
TVCHK_00:
    mov.b     @Sw_STA,r0l     ;Sw_STA ﾁｪｯｸ
    cmp.b     #1,r0l          ; Sw_STA=1 ならｶｳﾝﾄ
    beq       TVCHK_Sw
TVCHK_02:
    mov.b     @Peak_FG,r0l    ;Peak_FG ﾁｪｯｸ
    beq       TVCHK_Tx        ; 整流値ﾓｰﾄﾞならｶｳﾝﾄ
TVCHK_04:
    pop.l     er0
    rte

;-----  Sw ｳｪｲﾄｶｳﾝﾄ  ---------------
TVCHK_Sw:                     ;Sw_STA=1 : ﾀﾞﾌﾞﾙﾘｰﾄﾞ ｳｪｲﾄ中
    mov.b     @SwWaitCNT,r0l  ;SwWaitCNT ｶｳﾝﾄ
    inc.b     r0l             ; ｳｪｲﾄ時間=5msec
    cmp.b     #5,r0l          ;ｶｳﾝﾄｱｯﾌﾟ？
    bcs       TVCHK_Sw00      ; Cy=0 : ｶｳﾝﾄｱｯﾌﾟ
    mov.b     #2,r0l          ;Sw_STA 次のｽﾃｰｼﾞへ
    mov.b     r0l,@Sw_STA     ; Sw_STA=2
    xor.b     r0l,r0l         ;ｶｳﾝﾄ ｸﾘｱ
TVCHK_Sw00:
    mov.b     r0l,@SwWaitCNT  ;ｳｪｲﾄｶｳﾝﾀ ｸﾘｱ
    bra       TVCHK_02

;-----  整流値送信間隔ｶｳﾝﾄ  --------
TVCHK_Tx:
    mov.w     @Tx_CNT,r0      ;Tx_CNT ｶｳﾝﾄ
    inc.w     #1,r0
    cmp.w     #TxTiming,r0    ;1msec x TxTiming
    bcs       TVCHK_Tx00
    mov.b     #1,r0l          ;Tx_FG ｾｯﾄ
    mov.b     r0l,@Tx_FG
    xor.w     r0,r0           ;Tx_CNT ｸﾘｱ
TVCHK_Tx00:
    mov.w     r0,@Tx_CNT
    bra       TVCHK_04


;=========================================================
;          プログラム・スタート
;=========================================================
    .export         _main
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------------------------------
    bsr       INIT_PIO:16     ;PIO ｲﾆｼｬﾗｲｽﾞ
    bsr       INIT_ADC:16     ;A/D ｺﾝﾊﾞｰﾀ ｲﾆｼｬﾗｲｽﾞ
    bsr       INIT_TV:16      ;ﾀｲﾏ V INIT
    bsr       INIT_SCI:16     ;SCI3 ｲﾆｼｬﾗｲｽﾞ

    bsr       Calibration:16  ;ｷｬﾘﾌﾞﾚｰｼｮﾝ

    xor.l     er0,er0
    mov.l     er0,@AD_BUF     ;A/D 加算ﾊﾞｯﾌｧ ｸﾘｱ
    mov.w     r0,@absAD_data  ;絶対値化 A/D ﾊﾞｯﾌｧ ｸﾘｱ
    mov.w     r0,@AD_MAX      ;A/D 最高値 ｸﾘｱ
    mov.b     r0l,@Sw_STA     ;Sw ｽﾃｰﾀｽ ｸﾘｱ
    mov.b     r0l,@SwWaitCNT  ;Sw ｳｪｲﾄｶｳﾝﾀ ｸﾘｱ
    mov.b     r0l,@Sw_FG      ;Sw ﾌﾗｸﾞ ｸﾘｱ
    mov.b     r0l,@Tx_FG      ;送信ﾌﾗｸﾞ ｸﾘｱ
    mov.b     r0l,@Peak_FG    ;ﾋﾟｰｸ表示ﾌﾗｸﾞ ｸﾘｱ
    mov.w     r0,@Tx_CNT      ;送信ﾀｲﾐﾝｸﾞｶｳﾝﾀ ｸﾘｱ

    mov.w     #REC_CNST,r0    ;整流値算出回数 ｾｯﾄ
    mov.w     r0,@RAD_No

    andc      #H'7F,CCR       ;割り込みｲﾈｰﾌﾞﾙ

;-----  ﾒｲﾝﾙｰﾌﾟ  ------------------------------------------
MLoop:
    bsr       ADC:16          ;A/D 変換と平均化
    bsr       abs16:16        ;絶対値化
MLoop_00:
    mov.b     @Peak_FG,r0l    ;表示ﾓｰﾄﾞ判定
```

＜リスト 5-1 AD_Peak.src (1/4)＞

```
     bne       MLoop_10        ; Peak_FG=1 : ﾋﾟｰｸ値ﾓｰﾄﾞ
     bra       MLoop_12        ;         =0 : 整流値ﾓｰﾄﾞ

MLoop_02:
     bsr       SwRead:16       ;Sw ﾘｰﾄﾞ
     mov.b     @Sw_FG,r0l      ;Sw_FG ﾁｪｯｸ
     bne       MLoop_20        ; Sw_FG=1 : Sw 押された
MLoop_04:
     mov.b     @Tx_FG,r0l      ;送信ﾌﾗｸﾞ ﾁｪｯｸ
     bne       MLoop_30        ; Tx_FG=1 : 送信
MLoop_06:
     bra       MLoop

;-----  ﾋﾟｰｸ値ﾓｰﾄﾞ  ----------------
MLoop_10:
     bsr       Compare:16      ;ﾋﾟｰｸ値比較
     bra       MLoop_02

;-----  整流値ﾓｰﾄﾞ  ----------------
MLoop_12:
     bsr       rectify:16      ;整流値算出
     bra       MLoop_02

;-----  ｽｲｯﾁ  ----------------------
MLoop_20:
     xor.b     r0l,r0l         ;Sw_FG ｸﾘｱ
     mov.b     r0l,@Sw_FG

     mov.b     @Peak_FG,r0l    ;Peak_FG 反転
     not.b     r0l             ; Peak_FG=0 : 整流値ﾓｰﾄﾞ
     mov.b     r0l,@Peak_FG    ;         =1 : ﾋﾟｰｸ値ﾓｰﾄﾞ
     beq       MLoop_22
     xor.w     r0,r0           ;ﾋﾟｰｸ値ﾓｰﾄﾞに切り替わった時は
     mov.w     r0,@AD_MAX      ; 最高値をｸﾘｱする
MLoop_22:
     bra       MLoop_04

;-----  送信  ----------------------
MLoop_30:
     xor.b     r0l,r0l         ;送信ﾌﾗｸﾞ ｸﾘｱ
     mov.b     r0l,@Tx_FG

     mov.b     @Peak_FG,r0l    ;ﾋﾟｰｸﾎｰﾙﾄﾞﾌﾗｸﾞ ﾁｪｯｸ
     bne       MLoop_32
     mov.w     @absAD_data,r0  ;Peak_FG=0 : 整流値を表示
     bra       MLoop_34

MLoop_32:                      ;Peak_FG=1 : ﾋﾟｰｸ値を表示
     mov.w     @AD_MAX,r0
MLoop_34:
     bsr       HEX2BCD999:16   ;BCD 変換
     bra       TxD:16          ;送信
     bra       MLoop_06


;=========================================================
;          サブルーチン
;=========================================================
;---------------------------------------------------------
;          PIO ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_PIO:                      ;PIO ｲﾆｼｬﾗｲｽﾞ
     mov.b     #B'11111111,r0l ; P10-17 : 出力
     mov.b     r0l,@PCR1
     xor.b     r0l,r0l
     mov.b     r0l,@PDR1
     rts
```

```
;---------------------------------------------------------
;          A/D ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_ADC:                      ;A/D ｲﾆｼｬﾗｲｽﾞ
     mov.b     #B'00001001,r0l ; 単一ﾓｰﾄﾞ AN1 70 ｽﾃｰﾄ
     mov.b     r0l,@ADCSR
     rts

;---------------------------------------------------------
;          ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_TV:                       ;ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
     mov.b#B'01001011,r0l      ; ｲﾝﾀﾗﾌﾟﾄ CP-A ｲﾈｰﾌﾞﾙ
     mov.br0l,@TCRV0           ; ｺﾝﾍﾟｱﾏｯﾁ A ｸﾘｱ
     mov.b#B'11100011,r0l      ; 内部 CLK / 128 = 6.4usec
     mov.br0l,@TCRV1
     mov.b#D'157,r0l           ;ﾀｲﾑｺﾝｽﾀﾝﾄﾚｼﾞｽﾀ A
     mov.br0l,@TCORA           ; = 157 * 6.4usec = 1msec
     rts

;---------------------------------------------------------
;          SCI3 ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_SCI:
     bset      #1,@PMR1
     xor.b     r0l,r0l
     mov.b     r0l,@SCR3
     mov.b     r0l,@SMR
     mov.b     #BITR,r0l
     mov.b     r0l,@BRR
     mov.w     #WAIT_1B,r0
INITSCI_00:
     dec.w     #1,r0
     bne       INITSCI_00
     mov.b     #H'30,r0l
     mov.b     r0l,@SCR3
     rts

;---------------------------------------------------------
;          キャリブレーション
;---------------------------------------------------------
Calibration:
     mov.w     #D'8192,r3      ;加算回数 : D'8192=2 の 13 乗
     xor.l     er4,er4         ;ER4=加算ﾊﾞｯﾌｧ
Calibration_00:
     bsr       ADC:16          ;ﾉｲｽﾞ除去された 1 変換ﾃﾞｰﾀ->R0
     xor.w     e0,e0
     add.l     er0,er4         ;加算
     dec.w     #1,r3
     bne       Calibration_00
     shll.l    er4             ;平均化
     shll.l    er4
     shll.l    er4
     mov.w     e4,@SilentLv_BUF

     rts

;---------------------------------------------------------
;          A/D 変換
;---------------------------------------------------------
ADC:
     orc       #H'80,CCR       ;割り込みﾃﾞｨｲﾈｰﾌﾞﾙ

     xor.l     er0,er0         ;AD_BUF ｸﾘｱ
     mov.l     er0,@AD_BUF
     mov.w     #AVE_CNST,r2    ;R2 = 加算回数
     mov.w     r2,e2
```

<リスト 5-1 AD_Peak.src (2/4)>

```
ADC_00:
     bset      #5,@ADCSR       ;変換開始
ADC_02:
     btst      #7,@ADCSR       ;変換終了 ?
     beq       ADC_02
     bclr      #7,@ADCSR       ;変換終了

     mov.w     @ADDRB,r0       ;A/D 値ﾘｰﾄﾞ
     xor.w     e0,e0
     mov.l     @AD_BUF,er1
     add.l     er0,er1         ;平均化の為の加算
     mov.l     er1,@AD_BUF

     dec.w     #1,r2           ;加算終了 ?
     bne       ADC_00

     andc      #H'7F,CCR       ;割り込みｲﾈｰﾌﾞﾙ

;-----  ｼﾌﾄによる平均化  -----------;平均化回数=16 回
     mov.l     @AD_BUF,er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     mov.w     e0,r0
     mov.w     r0,@AD_data     ;R0=平均化 A/D 値
     rts

;---------------------------------------------------------
;          絶対値化
;---------------------------------------------------------
abs16:
     mov.w     @AD_data,r0     ;R0=平均化 A/D 値
     mov.w     @SilentLv_BUF,e0      ;E0=無音時ﾚﾍﾞﾙ

     sub.w     e0,r0           ;山波形?
     bcc       abs16_00        ; 山波形ならそのまま
     not.w     r0              ;谷波形
     inc.w     #1,r0           ; 波形反転
abs16_00:
     mov.w     r0,@absAD_data
     rts

;---------------------------------------------------------
;          ｽｲｯﾁﾘｰﾄﾞ
;---------------------------------------------------------
SwRead:
     mov.b     @Sw_STA,r0l
     cmp.b     #0,r0l
     beq       SwRead_STA0
     cmp.b     #1,r0l
     beq       SwRead_STA1
     cmp.b     #2,r0l
     beq       SwRead_STA2

     xor.b     r0l,r0l
     mov.b     r0l,@Sw_STA
     mov.b     r0l,@Sw_FG
     rts

;----- 1st ﾘｰﾄﾞ --------------------
SwRead_STA0:                   ;ｽﾃｰﾀｽ=0
     mov.b     @PDRB,r0l       ;Sw ﾁｪｯｸ PB ﾘｰﾄﾞ
     not.b     r0l             ;反転
     and.b     #B'01110000,r0l ;Sw = bit6-4
     beq       SwRead_off      ;EQ = 押されていない
     mov.b     r0l,@Sw_1st     ;ﾀﾞﾌﾞﾙﾘｰﾄﾞの為 退避
     mov.b     #1,r0l          ;次のｽﾃｰﾀｽへ
     mov.b     r0l,@Sw_STA     ; Sw_STA=1
     rts

;----- ｳｪｲﾄ ------------------------
SwRead_STA1:                   ;ｽﾃｰﾀｽ=1
     rts                       ;ﾀｲﾏ V によるｶｳﾝﾄ中

;----- 2nd ﾘｰﾄﾞ --------------------
SwRead_STA2:                   ;ｽﾃｰﾀｽ=2
     xor.b     r0l,r0l         ;Sw_STA ｸﾘｱ
     mov.b     r0l,@Sw_STA
     mov.b     @PDRB,r0l       ;Sw ﾁｪｯｸ PB ﾘｰﾄﾞ
     not.b     r0l             ;反転
     and.b     #B'01110000,r0l ;Sw=bit6-4
     beq       SwRead_off      ;EQ=押されていない
     mov.b     @Sw_1st,r1l
     cmp.b     r0l,r1l         ;1st ﾘｰﾄﾞと同じ?
     bne       SwRead_off
;----- 入力確定 --------------------
     mov.b     #1,r0l
     btst      #4,r1l          ;Bit4=Sw1 : ROL=1
     bne       SwRead_on
     inc.b     r0l
     btst      #5,r1l          ;Bit5=Sw2 : ROL=2
     bne       SwRead_on
     inc.b     r0l
     btst      #6,r1l          ;Bit6=Sw3 : ROL=3
     beq       SwRead_off
;----- 入力有り --------------------
SwRead_on:
;----- ﾜﾝｼｮｯﾄ動作 ------------------
     mov.b     @Sw_No,r0h      ;押し続けは無視する
     cmp.b     r0l,r0h         ;前回と一致？
     bne       SwRead_on1
     xor.b     r0l,r0l         ; 押し続け : Sw_FG=0
     bra       SwRead_FS
;----- Sw 番号確定 ------------------
SwRead_on1:
     mov.b     r0l,@Sw_No      ;Sw_No ｾｯﾄ
     mov.b     #1,r0l          ;Sw_FG = 1
     bra       SwRead_FS

;----- 入力無し --------------------
SwRead_off:
     xor.b     r0l,r0l
     mov.b     r0l,@Sw_No      ;ｽｲｯﾁ番号ｸﾘｱ
SwRead_FS:
     mov.b     r0l,@Sw_FG
     rts

;---------------------------------------------------------
;          最高値の比較と更新
;---------------------------------------------------------
Compare:
     mov.w     @absAD_data,r1  ;今までの最高値との比較
     mov.w     @AD_MAX,r0
     cmp.w     r1,r0
     bcc       Compare_00      ;Cy=0 : 未更新
     mov.w     r1,@AD_MAX      ;Cy=1 : 最高値更新
     mov.b     #1,r0l          ;送信ﾌﾗｸﾞｾｯﾄ
     mov.b     r0l,@Tx_FG
Compare_00:
     rts
```

＜リスト 5-1 AD_Peak.src (3/4)＞

```
;---------------------------------------------------------
;          整流
;---------------------------------------------------------
rectify:
     mov.w     @absAD_data,r0  ;ER0 = 絶対値
     xor.w     e0,e0
     mov.l     @RAD_BUF,er4    ;ER4 = 整流値加算ﾃﾞｰﾀ
     add.l     er0,er4         ;加算
     mov.l     er4,@RAD_BUF
     mov.w     @RAD_No,r0      ;REC_CNST 回 加算する
     dec.w     #1,r0
     beq       rectify_10
     mov.w     r0,@RAD_No      ;加算続行
     bra       rectify_end

rectify_10:                    ;加算終了
     mov.l     @RAD_BUF,er4
     shll.l    er4             ;平均化 (512)
     shll.l    er4
     shll.l    er4
     shll.l    er4
     shll.l    er4
     shll.l    er4
     shll.l    er4
     mov.w     e4,@RAD_data    ;整流値 get
     mov.w     #REC_CNST,r0    ;整流値加算回数 ｾｯﾄ
     mov.w     r0,@RAD_No
     xor.l     er0,er0         ;整流値加算ﾊﾞｯﾌｧ ｸﾘｱ
     mov.l     er0,@RAD_BUF

;-----  時間計測用ﾎﾟｰﾄ出力  --------
     mov.b     @PDR1,r1l       ;P30 変換時間計測用出力
     xor.b     #H'01,r1l       ; 出力反転
     mov.b     r1l,@PDR1

rectify_end:
     rts

;---------------------------------------------------------
;          BCD 変換
;---------------------------------------------------------
;R0 = 元ﾃﾞｰﾀ(D'999 ﾏﾃﾞ)
HEX2BCD999:
     mov.b     #D'100,r1l
     divxu.b   r1l,r0          ;A/D 値 % 100
     mov.b     r0l,@AD_data3   ;100 位 ｾｯﾄ

     mov.b     r0h,r0l         ;余りを R0L に移動
     xor.b     r0h,r0h         ; R0 = 余り
     mov.b     #D'10,r1l
     divxu.b   r1l,r0          ;余り % 10
     mov.b     r0l,@AD_data2   ;10 位 ｾｯﾄ

     mov.b     r0h,@AD_data1   ;余り = 1 位 ｾｯﾄ

     rts

;---------------------------------------------------------
;          A/D 送信
;---------------------------------------------------------
TxD:
     mov.b     #CL,r0l
     bsr       SEND_SCI:16     ;画面ｸﾘｱ

     mov.b     @AD_data3,r0l
     bsr       HEX2ASCII:16
     bsr       SEND_SCI:16     ;100 位送信

     mov.b     @AD_data2,r0l
     bsr       HEX2ASCII:16
     bsr       SEND_SCI:16     ;10 位送信

     mov.b     @AD_data1,r0l
     bsr       HEX2ASCII:16
     bsr       SEND_SCI:16     ;1 位送信

     mov.b     #CR,r0l
     bsr       SEND_SCI:16     ;改行送信
TxD_end:
     rts

;---------------------------------------------------------
;          SEND_SCI : 送信
;---------------------------------------------------------
SEND_SCI:
     btst      #7,@SSR
     beq       SEND_SCI
     mov.b     r0l,@TDR
     rts

;---------------------------------------------------------
;          HEX -> ASCII 変換
;---------------------------------------------------------
HEX2ASCII:                     ;R0L=ﾃﾞｰﾀ(bit4-7=0)
     add.b     #H'30,r0l       ;数字ﾍｱｽｷｰ変換
     cmp.b     #H'3A,r0l
     bcs       HEX2ASCII_00    ;A-F?
     add.b     #H'07,r0l       ;A-F ﾍｱｽｷｰ変換
HEX2ASCII_00:
     rts


;=========================================================
;          データ・エリア
;=========================================================
     .section  D,data,locate=H'FD00
;-----  A/Dｺﾝﾊﾞｰﾀ  ----------------------------------------
SilentLv_BUF:  .res.w    1     ;零点値
AD_BUF:        .res.l    1     ;A/D 加算ﾊﾞｯﾌｧ
absAD_data:    .res.w    1     ;絶対値化 A/D 値
AD_MAX:        .res.w    1     ;A/D 最高値
RAD_BUF:       .res.w    1     ;整流値加算ﾊﾞｯﾌｧ
RAD_data:      .res.w    1     ;整流値ﾃﾞｰﾀ
RAD_No:        .res.w    1     ;整流値算出回数
AD_data:                       ;BCD 化 A/D 値
AD_data1:      .res.b    1
AD_data2:      .res.b    1
AD_data3:      .res.b    1
     .align    2
;-----  Sw  ----------------------------------------------
Sw_STA:        .res.b    1     ;ｽﾃｰﾀｽ
SwWaitCNT:     .res.b    1     ;ｳｪｲﾄ ｶｳﾝﾀ
Sw_1st:        .res.b    1     ;初回に押されたｽｲｯﾁ状態
Sw_FG:         .res.b    1     ;ｽｲｯﾁ ﾌﾗｸﾞ 1=押された
Sw_No:         .res.b    1     ;ｽｲｯﾁ No,
     .align    2
;-----  送信  ----------------------------------------------
Tx_FG:         .res.b    1     ;送信ﾌﾗｸﾞ
Peak_FG:       .res.b    1     ;ﾋﾟｰｸ表示ﾌﾗｸﾞ
     .align    2
Tx_CNT:        .res.w    1     ;送信ﾀｲﾐﾝｸﾞ ｶｳﾝﾀ


;=========================================================
     .end
```

＜リスト 5-1 AD_Peak.src (4/4)＞

6 最大音量計のプログラム

おまけプログラムとして 5 章で作成したプログラムに手を加えて“最大音量計”を作成してみましょう。ピーク値の表示機能のみを使用し、誰が一番大きな声を出せるかを競う“大声コンテスト”のようなイベントで使われる最大音量を表示する機械のイメージでゲーム性を持たせます。表示は“■”で数字を作成し、見易さ及びインパクトを付けてみます。このプログラムはおまけですので A/D 処理部より他の処理が大半を占めています。しかし今まで味気の無いプログラムでも少し手を加えてるだけで面白味が生まれてきます。そんな例の一つとして参考までに収録します。

■プログラムの考え方

・概要

ある音量を基準にして測定の開始・停止を行い、過去の最大値と比較して勝ち負けを表示します。一定の音量レベルを設けその値より A/D 値が上回ったら計測開始、また計測中に一定時間そのレベルを下回ったら計測終了としています。このレベルを設けることで音（ここでは声）の入力の始まりと終わりを識別します（プログラム中では TriggerLV で定義されています）。識別レベルを一定時間下回ったら測定を終了して今までの最大音量（バッファ HighScore）と今回の最大値（バッファ AD_MAX）とを比較し、今までの最大音量を越えていたら HighScore を更新します。尚、計測に入る時や、ハイスコアの更新時など表示メッセージを替えてゲーム性を高めてみました。このモードも測定値の A/D 値は“■”の数字で表示します。スイッチ 1 をリセットスイッチに設定しましたので、ゲーム終了後スイッチ 1 で再スタートします。尚、HighScore はプログラム実行中はクリアされません。

・スイッチ

概要で述べた通りスイッチ 1 はゲームのリセットスイッチです。ゲーム終了後このスイッチを押すとゲームを再スタートします。尚、他のスイッチへの割付けはありません。

・表示

A/D 値の表示には“■”を使用して数字を表現しています。4×7 ブロックで一つの数字を構成し、それを 3 桁表示します。ハイパーターミナルはカーソルを上の行へ戻すことが出来ないので、一旦 RAM 上に表示イメージを展開してから一括送信・表示します。

■プログラム

次頁に最大音量計**“AD_MAXvoice”**のリストを掲載します。今回のプログラムはテーブルデータのボリュームが大きくなり H’E800～H’EFFF 番地では収まりきらなかったので、“■”の数字やメッセージなどのデータを H’F780 番地に割り付けてあります。

```
;---------------------------------------------------------
;
;   FILE        :AD_MAXvoice.src
;   DATE        :Thu, Aug 21, 2003
;   DESCRIPTION :Main Program
;   CPU TYPE    :H8/3687
;
;   This file is generated by Hitachi Project Generator (Ver.2.1)
;                 programed by Toyo-linx,Co.,Ltd. / Y.Furukawa.
;
;---------------------------------------------------------
; ■ 概要
;   マイクに入力された最大音量を競うゲームです。画面の指示に従
; ってマイクに向かって大きな声を出して下さい。ゲーム中はその時
; の最大音量を表示。ゲーム終了時に今の最大音量と今までの最大音
; 量（ハイスコア）、及び勝敗を表示します。
;   AN1 に入力された音声信号をノイズ除去・絶対値化しピーク値を
; 送信します。ﾉｲｽﾞ除去の為の平均化回数は 16 回で変換時間は
; (1/CLK)x(70max)x16 = 56μsec(max) ですが、実際には 1 変換時間
; の遅れ・平均化処理が加わるので約 110μsec(実測値)です。
; ■ スイッチ仕様
;   スイッチ１はゲーム終了時に押すことでゲームを再スタートし
; ます。この時今までのハイスコアは保持されています。その他のス
; イッチは割付け無しです。
;   スイッチはダブルリードを行いチャタリングを除去。また 押し
; た時のみ反応するようにワンショット処理も行っています。
; ■ VR の調整
;   入力する音に対してあまり値が変わらないようであればボリュ
; ームを上げて下さい。逆にピーク値（表示上で約 330）にすぐ達し
; てしまう場合は逆にボリュームを下げてください。若干小さめにセ
; ッティングするとよいでしょう。


;----- ｲﾝｸﾙｰﾄﾞﾌｧｲﾙ ------------------------------------------
    .include  "io3687F_equ.inc"    ;H8/3687 I/O 定義ﾌｧｲﾙ

;----- 定数の定義 --------------------------------------------
;-----  SCI3  ----------------------
MHz       .equ D'20           ;X=20MHz
BAUD      .equ 38400          ;9600,19200,38400
BITR      .equ (MHz*D'1000000)/(BAUD*D'32)-1
WAIT_1B   .equ (MHz*D'1000000)/D'6/BAUD

;-----  AD 変換  --------------------
AVE_CNST  .equ D'16           ;加算回数
Silent_Lv .equ H'01C2         ;無音時ﾚﾍﾞﾙ
TriggerLV .equ D'100          ;ｹﾞｰﾑ測定開始ﾚﾍﾞﾙ

;-----  ｱｽｷｰｺｰﾄﾞ  ---------------------------------------
BS        .equ H'08
TB        .equ H'09
LF        .equ H'0A
CL        .equ H'0C
CR        .equ H'0D


;=========================================================
;          割り込みプログラム
;=========================================================
    .export   INT_TimerV

INT_TimerV:
    push.l    er0
;-----  ﾀｲﾏV ｶｳﾝﾄ監視  -------------------------------------
TVCHK:
    bclr      #6,@TCSRV       ;ﾀｲﾏ V ｶｳﾝﾄｱｯﾌﾟ
TVCHK_00:
    mov.b     @Sw_STA,r0l     ;Sw_STA ﾁｪｯｸ
    cmp.b     #1,r0l          ; Sw_STA=1 ならｶｳﾝﾄ
    beq       TVCHK_Sw

TVCHK_02:
    mov.b     @Game_STA,r0l   ;ｹﾞｰﾑｽﾃｰﾀｽ
    cmp.b     #2,r0l          ; 測定中ならｶｳﾝﾄ
    beq       TVCHK_Game

TVCHK_04:
    pop.l     er0
    rte

;-----  Sw ｳｪｲﾄｶｳﾝﾄ  ---------------
TVCHK_Sw:                     ;Sw_STA=1 : ﾀﾞﾌﾞﾙﾘｰﾄﾞ ｳｪｲﾄ中
    mov.b     @SwWaitCNT,r0l  ;SwWaitCNT ｶｳﾝﾄ
    inc.b     r0l             ; ｳｪｲﾄ時間=5msec
    cmp.b     #5,r0l          ;ｶｳﾝﾄｱｯﾌﾟ？
    bcs       TVCHK_Sw00      ; Cy=0 : ｶｳﾝﾄｱｯﾌﾟ
    mov.b     #2,r0l          ;Sw_STA 次のｽﾃｰｼﾞへ
    mov.b     r0l,@Sw_STA     ; Sw_STA=2
    xor.b     r0l,r0l         ;ｶｳﾝﾄ ｸﾘｱ
TVCHK_Sw00:
    mov.b     r0l,@SwWaitCNT  ;ｳｪｲﾄｶｳﾝﾀ ｸﾘｱ
    bra       TVCHK_02

;-----  ｹﾞｰﾑ計測中ｶｳﾝﾄ  ------------
TVCHK_Game:
    mov.b     @GameCNT_FG,r0l
    bne       TVCHK_Game02

    mov.w     @Game_CNT,r0    ;Tx_CNT ｶｳﾝﾄ
    inc.w     #1,r0
    cmp.w     #D'2000,r0      ;1msec * D'2000 = 2sec
    bcs       TVCHK_Game00
    mov.b     #1,r0l          ;GameCNT_FG ｾｯﾄ
    mov.b     r0l,@GameCNT_FG
    xor.w     r0,r0           ;Tx_CNT ｸﾘｱ
TVCHK_Game00:
    mov.w     r0,@Game_CNT
TVCHK_Game02:
    bra       TVCHK_04


;=========================================================
;          プログラム・スタート
;=========================================================
    .export   _main
_main:
;-----  ｲﾆｼｬﾗｲｽﾞ  ------------------------------------------
    bsr       INIT_ADC:16     ;A/D ｺﾝﾊﾞｰﾀ ｲﾆｼｬﾗｲｽﾞ
    bsr       INIT_TV:16      ;ﾀｲﾏ V INIT
    bsr       INIT_SCI:16     ;SCI3 ｲﾆｼｬﾗｲｽﾞ

    xor.l     er0,er0
    mov.l     er0,@AD_BUF     ;A/D 加算ﾊﾞｯﾌｧ ｸﾘｱ
    mov.w     r0,@absAD_data  ;絶対値化 A/D ﾊﾞｯﾌｧ ｸﾘｱ
    mov.w     r0,@AD_MAX      ;A/D 最高値 ｸﾘｱ

    mov.b     r0l,@Sw_STA     ;Sw ｽﾃｰﾀｽ ｸﾘｱ
    mov.b     r0l,@SwWaitCNT  ;Sw ｳｪｲﾄｶｳﾝﾀ ｸﾘｱ
    mov.b     r0l,@Sw_FG      ;Sw ﾌﾗｸﾞ ｸﾘｱ

    mov.b     r0l,@Tx_FG      ;送信ﾌﾗｸﾞ ｸﾘｱ
    mov.w     r0,@Tx_CNT      ;送信ﾀｲﾐﾝｸﾞｶｳﾝﾀ ｸﾘｱ

    mov.b     r0l,@Game_STA   ;ｹﾞｰﾑｽﾃｰﾀｽ ｸﾘｱ
    mov.w     e0,@Game_CNT    ;ｹﾞｰﾑ計測時間ｶｳﾝﾀ ｸﾘｱ
    mov.b     r0l,@GameCNT_FG
    mov.w     r0,@HighScore   ;ﾊｲｽｺｱ ｸﾘｱ
    mov.b     r0l,@HS_FG      ;ﾊｲｽｺｱﾌﾗｸﾞ ｸﾘｱ
```

＜リスト 6-1　AD_MAXvoice.src (1/8)＞

```
      bsr       Calibration:16  ;ｷｬﾘﾌﾞﾚｰｼｮﾝ

      andc      #H'7F,CCR       ;割り込みｲﾈｰﾌﾞﾙ

;-----  ﾒｲﾝﾙｰﾌﾟ  ----------------------------------------
MLoop:
      bsr       ADC:16          ;A/D 変換と平均化
      bsr       abs16:16        ;整流
MLoop_00:
      bsr       SwRead:16       ;Sw ﾘｰﾄﾞ
      mov.b     @Sw_FG,r0l      ;Sw_FG ﾁｪｯｸ
      bne       MLoop_10        ; Sw_FG=1 : Sw 押された
MLoop_02:
      bsr       Game:16         ;ｹﾞｰﾑ進行と表示のｾｯﾄ

      mov.b     @Tx_FG,r0l      ;送信ﾌﾗｸﾞ ﾁｪｯｸ
      beq       MLoop_04        ; Tx_FG=1 : 送信
      xor.b     r0l,r0l         ;送信ﾌﾗｸﾞ ｸﾘｱ
      mov.b     r0l,@Tx_FG
      bsr       GameTxD:16      ;ｹﾞｰﾑﾓｰﾄﾞ用送信
MLoop_04:
      bra       MLoop

;-----  ｽｲｯﾁ  ----------------------------------
MLoop_10:
      xor.b     r0l,r0l         ;Sw_FG ｸﾘｱ
      mov.b     r0l,@Sw_FG
      bsr       SwAction:16     ;各ｽｲｯﾁ動作
      bra       MLoop_02



;=========================================================
;          サブルーチン
;=========================================================
;---------------------------------------------------------
;          ｹﾞｰﾑﾓｰﾄﾞ : 音量比較と表示ｾｯﾄ
;---------------------------------------------------------
Game:
      mov.b     @Game_STA,r0l   ;ｽﾃｰﾀｽで分岐
      cmp.b     #0,r0l
      beq       Game_00:16
      cmp.b     #1,r0l
      beq       Game_10:16
      cmp.b     #2,r0l
      beq       Game_20:16
      cmp.b     #3,r0l
      beq       Game_30:16
      rts

;-----  ｽﾃｰﾀｽ=0:ﾀｲﾄﾙ表示  ----------------------
Game_00:
      mov.w     #PMSTA0_MTBL,r0 ;ﾀｲﾄﾙﾒｯｾｰｼﾞﾃｰﾌﾞﾙ ｱﾄﾞﾚｽ ｾｯﾄ
      mov.w     r0,@MSG_ADR
      mov.b     #1,r0l          ;送信ﾌﾗｸﾞ ｾｯﾄ
      mov.b     r0l,@Tx_FG
      mov.b     r0l,@Game_STA   ;次のｽﾃｰﾀｽへ
      rts

;-----  ｽﾃｰﾀｽ=1:測定待ち  ----------------------
Game_10:
      bsr       Compare:16      ;ﾋﾟｰｸ値との比較
      mov.w     @absAD_data,r0
      cmp.w     #TriggerLV,r0   ;測定開始ﾚﾍﾞﾙに達したか?
      bcc       Game_14
Game_12:
      rts
Game_14:                        ;測定開始
      xor.w     r0,r0
      mov.w     r0,@Game_CNT
      mov.b     r0l,@GameCNT_FG
      mov.w     #PMSTA1_MTBL,r0 ;測定中ﾒｯｾｰｼﾞﾃｰﾌﾞﾙ ｱﾄﾞﾚｽ ｾｯﾄ
      mov.w     r0,@MSG_ADR
      xor.b     r0l,r0l         ;ﾋﾟｰｸ値更新ﾌﾗｸﾞ ｸﾘｱ
      mov.b     r0l,@ADCMP_FG
      mov.b     #1,r0l          ;送信ﾌﾗｸﾞ ｾｯﾄ
      mov.b     r0l,@Tx_FG
      mov.b     #2,r0l          ;次のｽﾃｰﾀｽへ
      mov.b     r0l,@Game_STA
      rts

;-----  ｽﾃｰﾀｽ=2:測定中  ------------------------
Game_20:
      mov.b     @GameCNT_FG,r0l ;一定時間 TriggerLV 以下だった
      bne       Game_24         ; 測定終了

      bsr       Compare:16      ;ﾋﾟｰｸ値との比較
      mov.w     @absAD_data,r0
      cmp.w     #TriggerLV,r0   ;ﾄﾘｶﾞﾚﾍﾞﾙに達していない場合は
      bcs       Game_22         ; Game_CNT はｸﾘｱしない
      xor.w     r0,r0           ;Game_CNT のｸﾘｱ
      mov.w     r0,@Game_CNT
      mov.b     @ADCMP_FG,r0l   ;ﾋﾟｰｸ値は更新された?
      beq       Game_22
      xor.b     r0l,r0l         ;ﾋﾟｰｸ値更新ﾌﾗｸﾞ ｸﾘｱ
      mov.b     r0l,@ADCMP_FG
      mov.w     #PMSTA1_MTBL,r0 ;ﾋﾟｰｸ値更新:表示
      mov.w     r0,@MSG_ADR
      mov.b     #1,r0l          ;送信ﾌﾗｸﾞ ｾｯﾄ
      mov.b     r0l,@Tx_FG
Game_22:
      rts

Game_24:
      mov.b     #3,r0l          ;次のｽﾃｰﾀｽへ
      mov.b     r0l,@Game_STA
      bra       Game_30

;-----  ｽﾃｰﾀｽ=3:測定終了  ----------------------
Game_30:
      mov.w     @AD_MAX,r0
      mov.w     @HighScore,r1
      cmp.w     r0,r1
      bcs       Game_32
      xor.w     r0,r0           ;ﾊｲｽｺｱﾌﾗｸﾞ ｸﾘｱ
      bra       Game_34
Game_32:
      mov.w     r0,@HighScore   ;ﾊｲｽｺｱ更新
      mov.b     #1,r0l          ;ﾊｲｽｺｱﾌﾗｸﾞ ｾｯﾄ
Game_34:
      mov.b     r0l,@HS_FG

      mov.w     #PMSTA21_MTBL,r0      ;終了ﾒｯｾｰｼﾞ ｱﾄﾞﾚｽ ｾｯﾄ
      mov.w     r0,@MSG_ADR
      mov.b     #1,r0l          ;送信ﾌﾗｸﾞ ｾｯﾄ
      mov.b     r0l,@Tx_FG
      rts

;---------------------------------------------------------
;          A/D ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
INIT_ADC:                       ;A/D ｲﾆｼｬﾗｲｽﾞ
      mov.b     #B'00001001,r0l ; 単一ﾓｰﾄﾞ AN1 70 ｽﾃｰﾄ
      mov.b     r0l,@ADCSR
      rts

;---------------------------------------------------------
;          ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
;---------------------------------------------------------
```

＜リスト 6-1　AD_MAXvoice.src (2/8)＞

```
INIT_TV:                        ;ﾀｲﾏ V ｲﾆｼｬﾗｲｽﾞ
     mov.b     #B'01001011,r0l ; ｲﾝﾀﾗﾌﾟﾄ CP-A ｲﾈｰﾌﾞﾙ
     mov.b     r0l,@TCRV0      ; ｺﾝﾍﾟｱﾏｯﾁ A ｸﾘｱ
     mov.b     #B'11100011,r0l ; 内部 CLK / 128 = 6.4usec
     mov.b     r0l,@TCRV1
     mov.b     #D'157,r0l      ;ﾀｲﾑｺﾝｽﾀﾝﾄﾚｼﾞｽﾀ A
     mov.b     r0l,@TCORA      ; = 157 * 6.4usec = 1msec
     rts

;-------------------------------------------------------------
;          SCI3 ｲﾆｼｬﾗｲｽﾞ
;-------------------------------------------------------------
INIT_SCI:
     bset      #1,@PMR1
     xor.b     r0l,r0l
     mov.b     r0l,@SCR3
     mov.b     r0l,@SMR
     mov.b     #BITR,r0l
     mov.b     r0l,@BRR
     mov.w     #WAIT_1B,r0
INITSCI_00:
     dec.w     #1,r0
     bne       INITSCI_00
     mov.b     #H'30,r0l
     mov.b     r0l,@SCR3
     rts

;-------------------------------------------------------------
;          ｷｬﾘﾌﾞﾚｰｼｮﾝ
;-------------------------------------------------------------
Calibration:
     mov.w     #D'8192,r3      ;加算回数 : D'8192=2 の 13 乗
     xor.l     er4,er4         ;ER4=加算ﾊﾞｯﾌｧ
Calibration_00:
     bsr       ADC:16          ;ﾉｲｽﾞ除去された 1 変換ﾃﾞｰﾀ->R0
     xor.w     e0,e0
     add.l     er0,er4         ;加算
     dec.w     #1,r3
     bne       Calibration_00
     shll.l    er4             ;平均化
     shll.l    er4
     shll.l    er4
     mov.w     e4,@SilentLv_BUF
     rts

;-------------------------------------------------------------
;          A/D 変換
;-------------------------------------------------------------
ADC:
     orc       #H'80,CCR       ;割り込みﾃﾞｨｲﾈｰﾌﾞﾙ

     xor.l     er0,er0         ;AD_BUF ｸﾘｱ
     mov.l     er0,@AD_BUF
     mov.w     #AVE_CNST,r2    ;R2 = 加算回数
     mov.w     r2,e2
ADC_00:
     bclr      #7,@ADCSR       ;変換終了
     bset      #5,@ADCSR       ;変換開始
ADC_02:
     btst      #7,@ADCSR       ;変換終了 ?
     beq       ADC_02
;    bclr      #7,@ADCSR       ;変換終了

     mov.w     @ADDRB,r0       ;A/D 値ﾘｰﾄﾞ
     xor.w     e0,e0
     mov.l     @AD_BUF,er1
     add.l     er0,er1         ;平均化の為の加算
     mov.l     er1,@AD_BUF
     dec.w     #1,r2           ;加算終了 ?
     bne       ADC_00

     andc      #H'7F,CCR       ;割り込みｲﾈｰﾌﾞﾙ

;-----  ｼﾌﾄによる平均化 -----------;平均化回数=16 回
     mov.l     @AD_BUF,er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     shll.l    er0
     mov.w     e0,r0
     mov.w     r0,@AD_data     ;R0=平均化 A/D 値
     rts

;-------------------------------------------------------------
;          絶対値化
;-------------------------------------------------------------
abs16:
     mov.w     @AD_data,r0     ;R0=平均化 A/D 値
     mov.w     @SilentLv_BUF,e0     ;E0=無音時ﾚﾍﾞﾙ
     sub.w     e0,r0           ;山波形?
     bcc       abs16_00        ; 山波形ならそのまま
     not.w     r0              ;谷波形
     inc.w     #1,r0           ; 波形反転
abs16_00:
     mov.w     r0,@RAD_data
     rts

;-------------------------------------------------------------
;          ｽｲｯﾁﾘｰﾄﾞ
;-------------------------------------------------------------
SwRead:
     mov.b     @Sw_STA,r0l
     cmp.b     #0,r0l
     beq       SwRead_STA0
     cmp.b     #1,r0l
     beq       SwRead_STA1
     cmp.b     #2,r0l
     beq       SwRead_STA2
     xor.b     r0l,r0l
     mov.b     r0l,@Sw_STA
     mov.b     r0l,@Sw_FG
     rts

;----- 1st ﾘｰﾄﾞ --------------------
SwRead_STA0:                    ;ｽﾃｰﾀｽ=0
     mov.b     @PDRB,r0l       ;Sw ﾁｪｯｸ PB ﾘｰﾄﾞ
     not.b     r0l             ;反転
     and.b     #B'01110000,r0l ;Sw = bit6-4
     beq       SwRead_off      ;EQ = 押されていない
     mov.b     r0l,@Sw_1st     ;ﾀﾞﾌﾞﾙﾘｰﾄﾞの為 退避
     mov.b     #1,r0l          ;次のｽﾃｰﾀｽへ
     mov.b     r0l,@Sw_STA     ; Sw_STA=1
     rts

;----- ｳｪｲﾄ ------------------------
SwRead_STA1:                    ;ｽﾃｰﾀｽ=1
     rts                        ;ﾀｲﾏ V によるｶｳﾝﾄ中

;----- 2nd ﾘｰﾄﾞ --------------------
SwRead_STA2:                    ;ｽﾃｰﾀｽ=2
     xor.b     r0l,r0l         ;Sw_STA ｸﾘｱ
     mov.b     r0l,@Sw_STA

     mov.b     @PDRB,r0l       ;Sw ﾁｪｯｸ PB ﾘｰﾄﾞ
     not.b     r0l             ;反転
```

＜リスト 6-1　AD_MAXvoice.src (3/8)＞

```
     and.b     #B'01110000,r0l ;Sw=bit6-4
     beq       SwRead_off      ;EQ=押されていない
     mov.b     @Sw_1st,r1l
     cmp.b     r0l,r1l         ;1st ﾘｰﾄﾞと同じ?
     bne       SwRead_off
;----- 入力確定 --------------------
     mov.b     #1,r0l
     btst      #4,r1l          ;Bit4=Sw1 : ROL=1
     bne       SwRead_on
     inc.b     r0l
     btst      #5,r1l          ;Bit5=Sw2 : ROL=2
     bne       SwRead_on
     inc.b     r0l
     btst      #6,r1l          ;Bit6=Sw3 : ROL=3
     beq       SwRead_off
;----- 入力有り --------------------
SwRead_on:
;----- ﾜﾝｼｮｯﾄ動作 ------------------
     mov.b     @Sw_No,r0h      ;押し続けは無視する
     cmp.b     r0l,r0h         ;前回と一致？
     bne       SwRead_on1
     xor.b     r0l,r0l         ; 押し続け : Sw_FG=0
     bra       SwRead_FS

;----- Sw 番号確定 ------------------
SwRead_on1:
     mov.b     r0l,@Sw_No      ;Sw_No ｾｯﾄ
     mov.b     #1,r0l          ;Sw_FG = 1
     bra       SwRead_FS

;----- 入力無し --------------------
SwRead_off:
     xor.b     r0l,r0l
     mov.b     r0l,@Sw_No      ;ｽｲｯﾁ番号ｸﾘｱ
SwRead_FS:
     mov.b     r0l,@Sw_FG
     rts

;-------------------------------------------------------------
;          スイッチ動作
;-------------------------------------------------------------
SwAction:
     mov.b     @Sw_No,r0l
     cmp.b     #1,r0l
     beq       SwAction_10
     cmp.b     #2,r0l
     beq       SwAction_20
     cmp.b     #3,r0l
     beq       SwAction_30
     rts

;-----  Sw_1  ----------------------
SwAction_10:
     mov.b     @Game_STA,r0l   ;ｹﾞｰﾑ終了?
     cmp.b     #4,r0l
     bne       SwAction_12
     xor.w     r0,r0
     mov.w     r0,@AD_MAX      ;最高値をｸﾘｱする
     mov.b     r0l,@Game_STA   ;ｽﾃｰﾀｽ=0 : ｹﾞｰﾑﾘｾｯﾄ
SwAction_12:
     rts
;-----  Sw_2  ----------------------
SwAction_20:
     rts

;-----  Sw_3  ----------------------
SwAction_30:
     rts
```

```
;-------------------------------------------------------------
;          最高値の比較と更新
;-------------------------------------------------------------
Compare:
     mov.w     @RAD_data,r1          ;今までの最高値との比較
     mov.w     @AD_MAX,r0
     cmp.w     r1,r0
     bcc       Compare_00            ;Cy=0 : 未更新
     mov.w     r1,@AD_MAX            ;Cy=1 : 最高値更新
     mov.b     #1,r0l                ;送信ﾌﾗｸﾞｾｯﾄ
     mov.b     r0l,@ADCMP_FG
Compare_00:
     rts

;-------------------------------------------------------------
;          ｹﾞｰﾑﾓｰﾄﾞ用送信
;-------------------------------------------------------------
GameTxD:
     mov.w     @MSG_ADR,r3           ;R3=ﾒｯｾｰｼﾞｱﾄﾞﾚｽ
     bsr       Tx_MSG:16             ;ﾒｯｾｰｼﾞの表示
     mov.b     @Game_STA,r0l         ;ｽﾃｰﾀｽ=0 : ﾄﾘｶﾞﾚﾍﾞﾙに達す
るまで
     cmp.b     #1,r0l                ; 表示しない
     beq       GameTxD_00
     mov.w     @AD_MAX,r0            ;最大値読み出し
     bsr       Make_BlockNo:16       ;最大値のﾌﾞﾛｯｸ数字表示ﾃﾞ
-ﾀ作成
     bsr       Tx_BlockNo:16         ;ﾌﾞﾛｯｸ数字の表示

     mov.b     @Game_STA,r0l         ;ｽﾃｰﾀｽ=3 : 測定終了時のみ
     cmp.b     #3,r0l                ; 追加ﾒｯｾｰｼﾞあり
     beq       GameTxD_10
GameTxD_00:
     rts

GameTxD_10:                          ;ｽﾃｰﾀｽ=3
     mov.w     #PMSTA22_MTBL,r3      ;R3=ﾒｯｾｰｼﾞｱﾄﾞﾚｽ
     bsr       Tx_MSG:16             ;ﾒｯｾｰｼﾞの表示
     mov.b     @HS_FG,r0l            ;ﾊｲｽｺｱ更新判定
     bne       GameTxD_12
     mov.w     #PMSTA23L_MTBL,r3     ;ﾊｲｽｺｱ更新ならず
     bra       GameTxD_14
GameTxD_12:
     mov.w     #PMSTA23H_MTBL,r3     ;ﾊｲｽｺｱ更新
GameTxD_14:
     xor.b     r0l,r0l               ;ﾊｲｽｺｱﾌﾗｸﾞ ｸﾘｱ
     mov.b     r0l,@HS_FG
     bsr       Tx_MSG:16             ;ﾒｯｾｰｼﾞの表示
     mov.w     #PMSTA24_MTBL,r3
     bsr       Tx_MSG:16             ;ﾒｯｾｰｼﾞの表示
     bsr       SEND_HSW              ;ﾊｲｽｺｱの表示(全角)
     mov.w     #PMSTA25_MTBL,r3
     bsr       Tx_MSG:16             ;ﾒｯｾｰｼﾞの表示
     mov.b     #4,r0l                ;ｽﾃｰﾀｽ=4 : ﾘｾｯﾄ待ち
     mov.b     r0l,@Game_STA
     rts

;-------------------------------------------------------------
;          BCD 変換
;-------------------------------------------------------------
;R0 = 元ﾃﾞｰﾀ(D'999 ﾏﾃﾞ)
HEX2BCD999:
     mov.b     #D'100,r1l
     divxu.b   r1l,r0          ;A/D 値 % 100
     mov.b     r0l,@AD_data3   ;100 位 ｾｯﾄ

     mov.b     r0h,r0l         ;余りを R0L に移動
     xor.b     r0h,r0h         ; R0 = 余り
```

＜リスト 6-1　AD_MAXvoice.src (4/8)＞

```
     mov.b     #D'10,r1l
     divxu.b   r1l,r0          ;余り % 10
     mov.b     r0l,@AD_data2   ;10 位 ｾｯﾄ

     mov.b     r0h,@AD_data1   ;余り = 1 位 ｾｯﾄ

     rts

;---------------------------------------------------------
;          A/D 送信
;---------------------------------------------------------
TxD:
     mov.b     #CL,r0l
     bsr       SEND_SCI:16     ;画面ｸﾘｱ

     mov.w     @RAD_data,r0
     bsr       Make_BlockNo:16
     bsr       Tx_BlockNo:16
     rts

;---------------------------------------------------------
;          SEND_SCI : 送信
;---------------------------------------------------------
SEND_SCI:
     btst      #7,@SSR
     beq       SEND_SCI
     mov.b     r0l,@TDR
     rts

;---------------------------------------------------------
;          HEX -> ASCII 変換
;---------------------------------------------------------
HEX2ASCII:                     ;R0L=ﾃﾞｰﾀ(bit4-7=0)
     add.b     #H'30,r0l       ;数字ﾍｱｽｷｰ変換
     cmp.b     #H'3A,r0l
     bcs       HEX2ASCII_00    ;A-F?
     add.b     #H'07,r0l       ;A-F ﾍｱｽｷｰ変換
HEX2ASCII_00:
     rts

;---------------------------------------------------------
;          ハイスコアの表示(全角)
;---------------------------------------------------------
SEND_HSW:
     mov.w     @HighScore,r0   ;ﾊｲｽｺｱ ﾘｰﾄﾞ
     bsr       HEX2BCD999:16   ;10 進数に変換
     mov.l     #DISP_BUF,er3   ;DISP_BUF に表示ﾃﾞｰﾀをｾｯﾄする
;----- 全角文字へ変換 --------------
     mov.b     @AD_data3,r0l
     bsr       Word_make_set   ; 100 位
     mov.b     @AD_data2,r0l
     bsr       Word_make_set   ;  10 位
     mov.b     @AD_data1,r0l
     bsr       Word_make_set   ;   1 位
;----- 表示 ------------------------
     mov.l     #DISP_BUF,er3
     mov.w     #6,r1
     bsr       TxMSG_00:16
     rts

;----- 全角数字の作成とﾊﾞｯﾌｧへのｾｯﾄ -------------------------
Word_make_set:
     mov.w     #H'824F,r1      ;H'824F=全角数字の０(零)
     add.b     r0l,r1l         ;０+n で任意の全角数字に変換
     mov.w     r1,@er3         ;ﾊﾞｯﾌｧにｾｯﾄ
     adds.l    #2,er3          ;ﾎﾟｲﾝﾀ+2(全角文字なので+2)
     rts
```

```
;---------------------------------------------------------
;          ﾌﾞﾛｯｸ数字の表示ﾃﾞｰﾀ作成
;---------------------------------------------------------
Make_BlockNo:                  ;R0=Data(HEX)
     bsr       HEX2BCD999:16   ;Decimal 変換
     bsr       Set_DISPBUF:16  ;ﾊﾞｯﾌｧﾍﾌﾞﾛｯｸ数字を配置
     rts

;---------------------------------------------------------
;          ﾌﾞﾛｯｸ数字の表示
;---------------------------------------------------------
Tx_BlockNo:
     mov.l     #DISP_BUF,er3
     mov.w     #D'217,r1
     bsr       TxMSG_00:16
     rts

;---------------------------------------------------------
;          大文字４桁の表示セット
;---------------------------------------------------------
Set_DISPBUF:                   ;ER0 = A/D 値(Decimal)
;----- set Space -------------------
     mov.w     #D'31*7,r1      ;ﾃﾞｰﾀ数 ｾｯﾄ (15word+1)*7line
     mov.w     r1,@DISP_CNT
     mov.l     #DISP_BUF,er3   ;ER3 = DISP_BUF Address
     mov.b     #H'20,r2l       ;R2L = Space(H'20)
SDB_SetSP:
     mov.b     r2l,@er3        ;Space 書き込み
     adds.l    #1,er3          ;ｱﾄﾞﾚｽ +1
     dec.w     #1,r1           ;ﾃﾞｰﾀ数 -1
     bne       SDB_SetSP
;----- set CR ----------------------
     mov.w     #D'31*7,r1      ;ﾃﾞｰﾀ数 ｾｯﾄ
     mov.w     r1,@DISP_CNT
     mov.l     #DISP_BUF,er3   ;ER3 = ﾊﾞｯﾌｧｱﾄﾞﾚｽ
     mov.b     #CR,r1l         ;R1l = CR
     mov.b     #7,r1h          ;R1H = ｶｳﾝﾄ
SDB_SetCR:
     add.l     #D'30,er3
     mov.b     r1l,@er3
     adds.l    #1,er3
     dec.b     r1h
     bne       SDB_SetCR
;----- Set  100 --------------------
     xor.l     er2,er2         ;ER2 = 0
     mov.b     @AD_data3,r2l
     shll.l    er2             ;er2 x 2
     mov.w     @(BNUM_ATBL,er2),r3
     xor.w     e3,e3           ;ER3 = BlockNumber ｱﾄﾞﾚｽ
     mov.l     #DISP_BUF,er2   ;ER2 = DISP_BUF ｱﾄﾞﾚｽ
     add.l     #D'2,er2        ;ER2 +12 : 開始位置ｾｯﾄ
     bsr       Set_BN:16       ;表示ﾃﾞｰﾀのｾｯﾄ
;----- Set   10 --------------------
     xor.l     er2,er2         ;ER2 = 0
     mov.b     @AD_data2,r2l
     shll.l    er2             ;er2 x 2
     mov.w     @(BNUM_ATBL,er2),r3
     xor.w     e3,e3           ;ER3 = BlockNumber ｱﾄﾞﾚｽ
     mov.l     #DISP_BUF,er2   ;ER2 = DISP_BUF ｱﾄﾞﾚｽ
     add.l     #D'12,er2       ;ER2 +12 : 開始位置ｾｯﾄ
     bsr       Set_BN:16       ;表示ﾃﾞｰﾀのｾｯﾄ

;----- Set    1 --------------------
     xor.l     er2,er2         ;ER2 = 0
     mov.b     @AD_data1,r2l
     shll.l    er2             ;er2 x 2
     mov.w     @(BNUM_ATBL,er2),r3
     xor.w     e3,e3           ;ER3 = BlockNumber ｱﾄﾞﾚｽ
     mov.l     #DISP_BUF,er2   ;ER2 = DISP_BUF ｱﾄﾞﾚｽ
```

＜リスト 6-1　AD_MAXvoice.src (5/8)＞

```
     add.l     #D'22,er2       ;ER2 +12 : 開始位置ｾｯﾄ
     bsr       Set_BN:16       ;表示ﾃﾞｰﾀのｾｯﾄ
     rts

;-------------------------------------------------------
;          ﾌﾞﾛｯｸﾅﾝﾊﾞｰ ｾｯﾄ
;-------------------------------------------------------
;ER0 = Can not use.
; R1 = ｺﾋﾟｰ総数
; E1 = 1ﾗｲﾝ数
;ER2 = DISP_BUF ｱﾄﾞﾚｽ
;ER3 = ｺﾋﾟｰ元 ｱﾄﾞﾚｽ
Set_BN:
     mov.w     @r3,r1          ; R1 = ﾃﾞｰﾀ数
     inc.w     #2,r3
SetBN_00:
     mov.w     #8,e1           ; E1 = 1ﾗｲﾝ ﾃﾞｰﾀ数
SetBN_02:
     mov.b     @er3+,r4l       ;(ER3) -> (ER2)
     mov.b     r4l,@er2        ;ER3+1
     adds.l    #1,er2          ;ER2+1
     dec.w     #1,r1           ;ﾃﾞｰﾀ数-1
     beq       SetBN_04
     dec.w     #1,e1           ;ﾗｲﾝ数 -1
     bne       SetBN_02
     add.l     #D'23,er2       ;DISP_BUF ｱﾄﾞﾚｽ + D'33
     bra       SetBN_00
SetBN_04:
     rts

;-------------------------------------------------------
;          ﾒｯｾｰｼﾞ送信
;-------------------------------------------------------
Tx_MSG:                        ;R3=ﾒｯｾｰｼﾞﾃｰﾌﾞﾙｱﾄﾞﾚｽ
     mov.w     @r3,r1          ;R1L=ﾃﾞｰﾀ数
     inc.w     #2,r3
TxMSG_00:
     mov.b     @r3,r0l         ;R0L=送信ﾃﾞｰﾀ
     bsr       SEND_SCI:16     ;送信
     inc.w     #1,r3           ;ｱﾄﾞﾚｽ+1
     dec.w     #1,r1           ;ﾃﾞｰﾀ数-1
     bne       TxMSG_00
     rts

;-------------------------------------------------------
;          画面のｸﾘｱ(CL 送信)
;-------------------------------------------------------
SEND_CL:
     mov.b     #CL,r0l
     bsr       SEND_SCI:16
     rts


;=======================================================
;          テーブルデータ・エリア
;=======================================================
     .section  D,data,locate=H'F780
;----- PMSTA0_MTBL : ﾋﾟｰｸﾒｰﾀｽﾀｰﾄﾒｯｾｰｼﾞ ----------------------
PMSTA0_MTBL:
     .data.w   PMSTA0_MTBLE-PMSTA0_MTBL-2
     .data.b   CL
     .sdata    "マイクに向かって大きな声で叫べ！！"<H'0D>
     .sdata    ""<H'0D>
     .data.b   CR
     .data.b   CR,CR,CR
     .sdata    "  ■■■■  ■■■■  ■■■■"<H'0D>
     .data.b   CR,CR,CR
PMSTA0_MTBLE:
     .align    2

;----- PMSTA1_MTBL : ﾋﾟｰｸﾒｰﾀ計測中ﾒｯｾｰｼﾞ ----------------------
PMSTA1_MTBL:
     .data.w   PMSTA1_MTBLE-PMSTA1_MTBL-2
     .data.b   CL
     .sdata    "マイクに向かって大きな声で叫べ！！"<H'0D>
     .sdata    "もっともっと大きな声で！！！"<H'0D>
     .data.b   CR
PMSTA1_MTBLE:
     .align    2

;----- PMSTA21_MTBL : ﾋﾟｰｸﾒｰﾀ計測終了ﾒｯｾｰｼﾞ_1 ----------------
PMSTA21_MTBL:
     .data.w   PMSTA21_MTBLE-PMSTA21_MTBL-2
     .data.b   CL
     .sdata    "測定終了！　おつかれさま！"<H'0D>
     .sdata    "只今の最大音量は... "<H'0D>
     .data.b   CR
PMSTA21_MTBLE:
     .align    2

;----- PMSTA22_MTBL : ﾋﾟｰｸﾒｰﾀ計測終了ﾒｯｾｰｼﾞ_2 ----------------
PMSTA22_MTBL:
     .data.w   PMSTA22_MTBLE-PMSTA22_MTBL-2
     .data.b   CR
     .sdata    "                    でした。"<H'0D>
PMSTA22_MTBLE:
     .align    2

;----- PMSTA23H_MTBL : ﾋﾟｰｸﾒｰﾀ計測終了ﾒｯｾｰｼﾞ_3H ﾊｲｽｺｱ更新 ----
PMSTA23H_MTBL:
     .data.w   PMSTA23H_MTBLE-PMSTA23H_MTBL-2
     .sdata    "ハイスコア更新！！"<H'0D>
     .data.b   CR
PMSTA23H_MTBLE:
     .align    2

;----- PMSTA23L_MTBL : ﾋﾟｰｸﾒｰﾀ計測終了ﾒｯｾｰｼﾞ_3L 更新ならず ---
PMSTA23L_MTBL:
     .data.w   PMSTA23L_MTBLE-PMSTA23L_MTBL-2
     .sdata    "残念。ハイスコア更新ならず... "<H'0D>
     .data.b   CR
PMSTA23L_MTBLE:
     .align    2

;----- PMSTA24_MTBL : ﾋﾟｰｸﾒｰﾀ計測終了ﾒｯｾｰｼﾞ_4 ----------------
PMSTA24_MTBL:
     .data.w   PMSTA24_MTBLE-PMSTA24_MTBL-2
     .sdata    "現在のハイスコアは "
PMSTA24_MTBLE:
     .align    2

;----- PMSTA25_MTBL : ﾋﾟｰｸﾒｰﾀ計測終了ﾒｯｾｰｼﾞ_5 ----------------
PMSTA25_MTBL:
     .data.w   PMSTA25_MTBLE-PMSTA25_MTBL-2
     .sdata    " です。"<H'0D>
     .sdata    "スイッチ１で再チャレンジ！"<H'0D>
PMSTA25_MTBLE:
     .align    2

;-----  ﾌﾞﾛｯｸﾅﾝﾊﾞｰ ｱﾄﾞﾚｽ ﾃｰﾌﾞﾙ  ----------------------------
BNUM_ATBL:
     .data.w   BlockNumber_0
     .data.w   BlockNumber_1
     .data.w   BlockNumber_2
     .data.w   BlockNumber_3
     .data.w   BlockNumber_4
     .data.w   BlockNumber_5
     .data.w   BlockNumber_6
```

＜リスト 6-1　AD_MAXvoice.src (6/8)＞

```
      .data.w   BlockNumber_7
      .data.w   BlockNumber_8
      .data.w   BlockNumber_9

;-----  ﾌﾞﾛｯｸﾅﾝﾊﾞｰ ﾃﾞｰﾀﾃｰﾌﾞﾙ  ------------------------------
BlockNumber_0:
      .data.w   BlockNumber_0E-BlockNumber_0-2
      .sdata    "  ■■  "
      .sdata    "■    ■"
      .sdata    "■    ■"
      .sdata    "■    ■"
      .sdata    "■    ■"
      .sdata    "■    ■"
      .sdata    "  ■■  "
BlockNumber_0E:
      .align    2

BlockNumber_1:
      .data.w   BlockNumber_1E-BlockNumber_1-2
      .sdata    "    ■  "
      .sdata    "  ■■  "
      .sdata    "    ■  "
      .sdata    "    ■  "
      .sdata    "    ■  "
      .sdata    "    ■  "
      .sdata    "  ■■■"
BlockNumber_1E:
      .align    2

BlockNumber_2:
      .data.w   BlockNumber_2E-BlockNumber_2-2
      .sdata    "  ■■  "
      .sdata    "■    ■"
      .sdata    "     ■"
      .sdata    "    ■  "
      .sdata    "  ■    "
      .sdata    "■      "
      .sdata    "■■■■"
BlockNumber_2E:
      .align    2

BlockNumber_3:
      .data.w   BlockNumber_3E-BlockNumber_3-2
      .sdata    "  ■■  "
      .sdata    "■    ■"
      .sdata    "     ■"
      .sdata    "  ■■  "
      .sdata    "     ■"
      .sdata    "■    ■"
      .sdata    "  ■■  "
BlockNumber_3E:
      .align    2

BlockNumber_4:
      .data.w   BlockNumber_4E-BlockNumber_4-2
      .sdata    "    ■  "
      .sdata    "  ■■  "
      .sdata    "■  ■  "
      .sdata    "■  ■  "
      .sdata    "■■■■"
      .sdata    "    ■  "
      .sdata    "    ■  "
BlockNumber_4E:
      .align    2
```

```
BlockNumber_5:
      .data.w   BlockNumber_5E-BlockNumber_5-2
      .sdata    "■■■  "
      .sdata    "■      "
      .sdata    "■      "
      .sdata    "■■■  "
      .sdata    "      ■"
      .sdata    "■    ■"
      .sdata    "  ■■  "
BlockNumber_5E:
      .align    2

BlockNumber_6:
      .data.w   BlockNumber_6E-BlockNumber_6-2
      .sdata    "  ■■  "
      .sdata    "■    ■"
      .sdata    "■      "
      .sdata    "■■■  "
      .sdata    "■    ■"
      .sdata    "■    ■"
      .sdata    "  ■■  "
BlockNumber_6E:
      .align    2

BlockNumber_7:
      .data.w   BlockNumber_7E-BlockNumber_7-2
      .sdata    "■■■■■"
      .sdata    "     ■"
      .sdata    "     ■"
      .sdata    "   ■  "
      .sdata    " ■    "
      .sdata    " ■    "
      .sdata    " ■    "
BlockNumber_7E:
      .align    2

BlockNumber_8:
      .data.w   BlockNumber_8E-BlockNumber_8-2
      .sdata    "  ■■  "
      .sdata    "■    ■"
      .sdata    "■    ■"
      .sdata    "  ■■  "
      .sdata    "■    ■"
      .sdata    "■    ■"
      .sdata    "  ■■  "
BlockNumber_8E:
      .align    2

BlockNumber_9:
      .data.w   BlockNumber_9E-BlockNumber_9-2
      .sdata    "  ■■  "
      .sdata    "■    ■"
      .sdata    "■    ■"
      .sdata    "  ■■■"
      .sdata    "     ■"
      .sdata    "■    ■"
      .sdata    "  ■■  "
BlockNumber_9E:
      .align    2

;////  ﾌﾞﾛｯｸﾅﾝﾊﾞｰを展開する為のﾊﾞｯﾌｧ  ////////////////////
      .org H'FC00
DISP_BUF:       .res.b    H'E0 ;ﾌﾞﾛｯｸﾅﾝﾊﾞｰ展開ﾊﾞｯﾌｧ
```

＜リスト 6-1　AD_MAXvoice.src (7/8)＞

```
;=======================================================
;          データ・エリア
;=======================================================
     .org H'FD00

;-----  ｹﾞｰﾑ  ----------------------------------------
Game_STA:       .res.b    1     ;ｹﾞｰﾑｽﾃｰﾀｽ
     .align     2
Game_CNT:       .res.w    1     ;ｹﾞｰﾑ計測時間 ｶｳﾝﾀ
GameCNT_FG:     .res.b    1     ;
     .align     2
HighScore:      .res.w    1     ;ﾊｲｽｺｱ
HS_FG:          .res.b    1     ;ﾊｲｽｺｱ更新 ﾌﾗｸﾞ
     .align     2

;-----  A/Dｺﾝﾊﾞｰﾀ  ------------------------------------
SilentLv_BUF:   .res.w    1     ;零点値
AD_BUF:         .res.l    1     ;A/D 加算ﾊﾞｯﾌｧ
absAD_data:     .res.w    1     ;絶対値化 A/D 値
AD_MAX:         .res.w    1     ;A/D 最高値
AD_data:                        ;BCD 化 A/D 値
AD_data1:       .res.b    1
AD_data2:       .res.b    1
AD_data3:       .res.b    1
ADCMP_FG:       .res.b    1     ;ﾋﾟｰｸ値更新 ﾌﾗｸﾞ
     .align     2
```

```
;-----  Sw  ------------------------------------------
Sw_STA:         .res.b    1     ;ｽﾃｰﾀｽ
SwWaitCNT:      .res.b    1     ;ｳｪｲﾄ ｶｳﾝﾀ
Sw_1st:         .res.b    1     ;初回に押されたｽｲｯﾁ状態
Sw_FG:          .res.b    1     ;ｽｲｯﾁ ﾌﾗｸﾞ 1=押された
Sw_No:          .res.b    1     ;ｽｲｯﾁ No,
     .align     2

;-----  送信  ----------------------------------------
Tx_FG:          .res.b    1     ;送信ﾌﾗｸﾞ
     .align     2
Tx_CNT:         .res.w    1     ;送信ﾀｲﾐﾝｸﾞ ｶｳﾝﾀ
MSG_ADR:        .res.w    1     ;ﾒｯｾｰｼﾞﾃｰﾌﾞﾙ ｱﾄﾞﾚｽ

;-----  表示  ----------------------------------------
DISP_CNT:       .res.w    1     ;表示ﾃﾞｰﾀ数


;=======================================================
          .end
```

＜リスト 6-1　AD_MAXvoice.src (8/8)＞

■SCI3 のイニシャライズと送信

SCI3 はパソコンなどと非同期通信を行うシリアルコミュニケーションインターフェースです。ハイパーターミナルとの通信もこの SCI3 を用いています。ここでは SCI3 のイニシャライズの各意味と送信プログラムについて解説します。

・イニシャライズ

```
;-----  SCI3 ｲﾆｼｬﾗｲｽﾞ  ----------------------------------------------------
MHz       .equ  D'20              ;X=20MHz
BAUD      .equ  38400             ;38400bps
BITR      .equ  (MHz*D'1000000)/(BAUD*D'32)-1
WAIT_1B   .equ  (MHz*D'1000000)/D'6/BAUD

INIT_SCI:
      bset      #1,@PMR1      ①
      xor.b     r0l,r0l
      mov.b     r0l,@SCR3     ②
      mov.b     r0l,@SMR      ③
      mov.b     #BITR,r0l
      mov.b     r0l,@BRR      ④
      mov.w     #WAIT_1B,r0   ⑤
INITSCI_00:
      dec.w     #1,r0
      bne       INITSCI_00
      mov.b     #H'30,r0l     ⑥
      mov.b     r0l,@SCR3
      rts
```

＜リスト 6-1　SCI3 イニシャライズ＞

上記リストは 2 章で用いた SCI3 のイニシャライズ部です。各設定の意味を説明していきます。

① ポートの設定
ポートモードレジスタ 1(PMR1)の bit0 を 1 にすると P22/TxD 端子を TxD 端子に指定されます。

② 送受信動作の停止
シリアルコントロールレジスタ 3(SCR3)の RE(bit4)と TE(bit5)を 0 にして送受信動作を停止します。

③ シリアルモードレジスタ(SMR)の設定

SMR

7	6	5	4	3	2	1	0
COM	**CHR**	**PE**	**PM**	**STOP**	**MP**	**CKS1**	**CKS0**
0	0	0	0	0	0	0	0

COM・・・・・コミュニケーションモード　0=調歩同期式モード
CHR・・・・・キャラクタレングス　0=データ長 8bit
PE ・・・・・パリティイネーブル　0=ノンパリティ
PM ・・・・・パリティモード　ノンパリティなので関係なし
STOP ・・・・ストップビットレングス 0=1 ストップビット
MP ・・・・・マルチプロセッサ　0=ディセーブル
CKS1-0 ・・・クロックセレクト　00=φクロック(n=0)

＜設定内容＞
調歩同期、データ長 8bit、ノンパリティ、ストップ 1bit、
クロックソース φクロック(n=0)

④ ビットレートレジスタ(BBR)の設定
ボーレートを設定します。設定値は.equ で定義されている BITR です。BITR の計算式については“H8/3687 シリーズ　ハードウェアマニュアル”を参照して下さい。

⑤ BBR 設定後は少なくとも 1 ビット期間（38400bps なら、約 26μsec）待ってから、TE・RE をセットします。

⑥ 送受信のイネーブル
SCR3 の RE(bit4)、TE(bit5)を 1 にして送受信をイネーブルにします。

・送信

```
;-----  SEND_SCI : 送信  ------------------------------------------------
SEND_SCI:
        btst    #7,@SSR         ①
        beq     SEND_SCI
        mov.b   r0l,@TDR        ②
        rts
```

＜リスト 6-2　SCI3 送信＞

上記リストは 2 章で用いた SCI3 の送信部です。各動作の説明をしていきます。

① 送信可能かをチェック
シリアルステータスレジスタ(SSR)の bit7、トランスミットデータレジスタエンプティ(TDRE)をチェックし、トランスミットデータレジスタが空いているかチェックします。

② 送信
トランスミットデータレジスタ(TDR)に送信データをセットし送信を行います。

尚、より詳しい情報は“H8/3687 シリーズ　ハードウェアマニュアル”の 16 章「シリアルコミュニケーションインターフェース 3(SCI3)」を参照ください。

■プログラム中の擬似命令解説

.include “＜ファイル名＞”

指定したファイルをソース内に取り込みます。テキスト内のプログラムでは内蔵モジュールアドレス定義ファイル“io3687F_equ.inc”を取り込むのに使用しています。

.export ＜シンボル＞

ソース内で定義したシンボルが別のソースから参照される場合に宣言します。

.import ＜シンボル＞

別ソースの.export で定義したシンボルを参照するのに使用します。

＜ラベル＞ .assignc “＜文字列＞”
.aif “¥&＜ラベル＞” EQ “＜文字列＞”
.aendi

.assignc で定義したラベルの文字列が.aif のラベル及び文字列と同じなら、.aif～.aendi をビルドします。もし、文字列が違う場合はビルドされません。ビルドするソースを選択したい時に使用します。また EQ を NE にすると文字列が違い場合にビルドされます。

.align ＜境界調整値＞

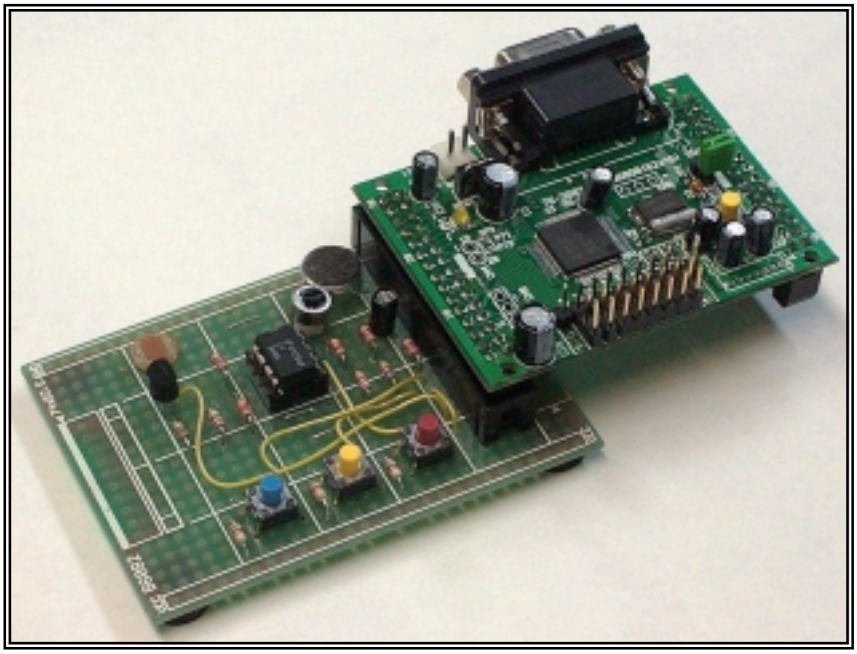
セクション内のアドレスを境界調整値の倍数に補正します。バッファなどをレジスタ間接でアクセスする際アドレスは偶数でなければなりません。そこでフラグやバイトサイズのバッファの後にこの擬似命令を入れておくとアドレスが偶数になるようにアセンブラが調整してくれます。境界調整値は 2 です。

.end

ソースプログラムの終わりを示します。

※擬似命令のより詳しい詳細については“H8S、H8/300 シリーズ　C/C++コンパイラ、アセンブラ、最適化リンケージエディタ　ユーザーズマニュアル”を参照して下さい。

TK-3687miniで使うための
ハードの組立て



ハードの組み立て

各プログラムを作成する前に付属のユニバーサル基板にハードを組み上げます。プリント基板と違いユニバーサル基板は全ての配線を自分で結線しなければなりません。次ページの回路図を見ながら部品をハンダ付けしていく事によって、ハードの構成を理解してみましょう。

まず、工具・部品の確認を行いましょう。下記の部品表と照らし合わせて全ての部品が揃っている事を確認して下さい。

■工具

ハンダごて、ハンダ、ニッパ、ワイヤストリッパ、ピンセット

■部品

TK-3687miniオプション　ADコンバータ　部品表	全数:	1

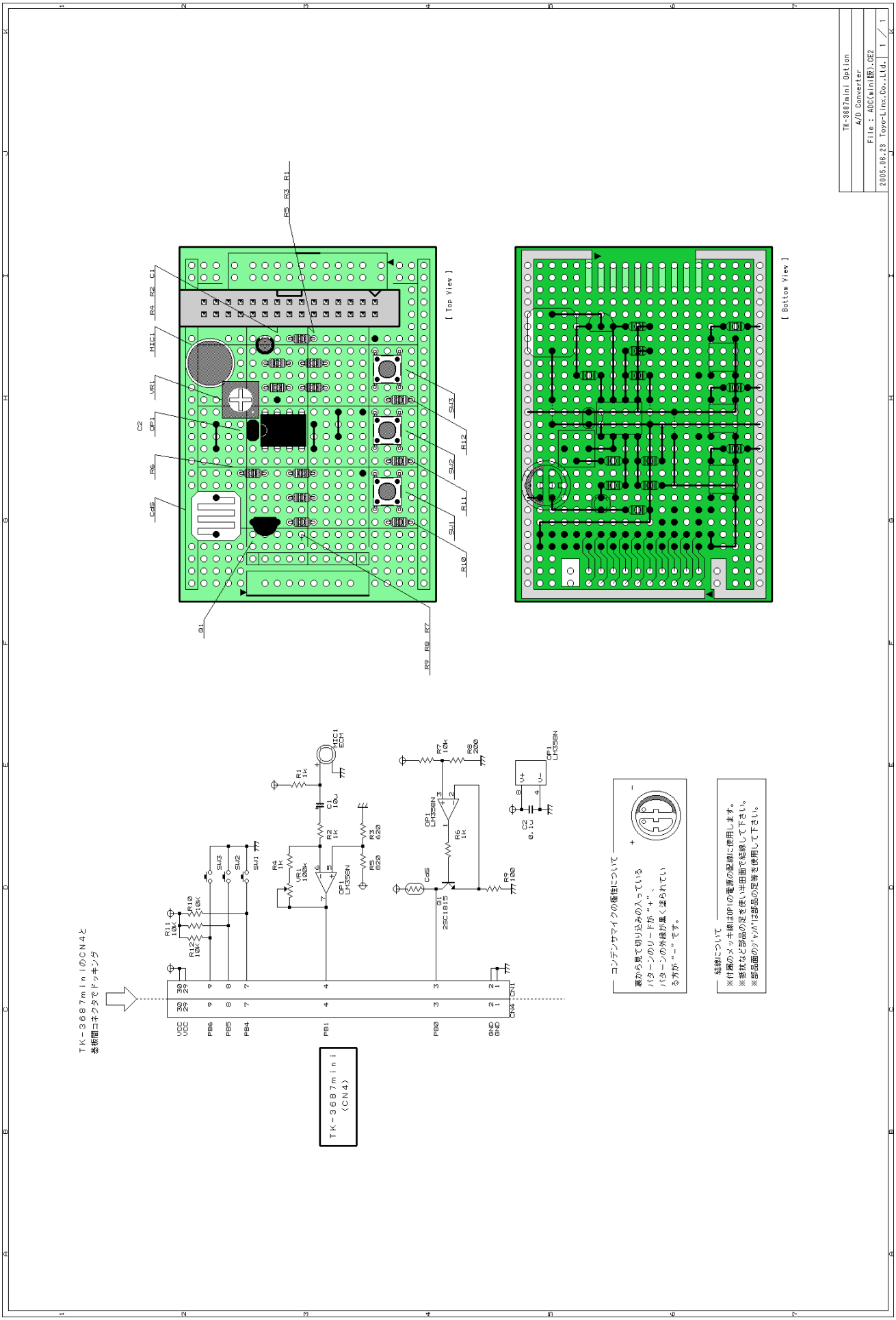
	部品番号	型名，規格	メーカー	数量	全数	備考
1	OP1	LM358N		1	1	単電源デュアルオペアンプ
2	Q1	2SC1815		1	1	
3	CdS			1	1	光センサ
4	ECM			1	1	コンデンサマイク
5	R9	100Ω		1	1	
6	R8	200Ω		1	1	
7	R1,2,4,6	1kΩ		4	4	
8	R3	620Ω		1	1	
9	R5	820Ω		1	1	
10	R7,10,11,12	10kΩ		4	4	
11	VR1	100kΩ		1	1	
12	C2	0.1μF　（積セラ）		1	1	
13	C1	10μF/16V　（電解）		1	1	
14	SW1,2,3			3	3	
15	CN1	HIF3FC-30PA-2.54DSA		1	1	
16	基板	B6082	東洋リンクス	1	1	
17	ゴム足			4	4	
18	ラッピングケーブル	50cm		1	1	結線用
19	メッキ線					Vcc,GND等半田面結線用　※1
20						
21						
22						

※相当品を使用する場合があります。
※1　ラッピングケーブルの被覆を剥し2本をよじって使用します。また抵抗やコンデンサの足も流用できます。

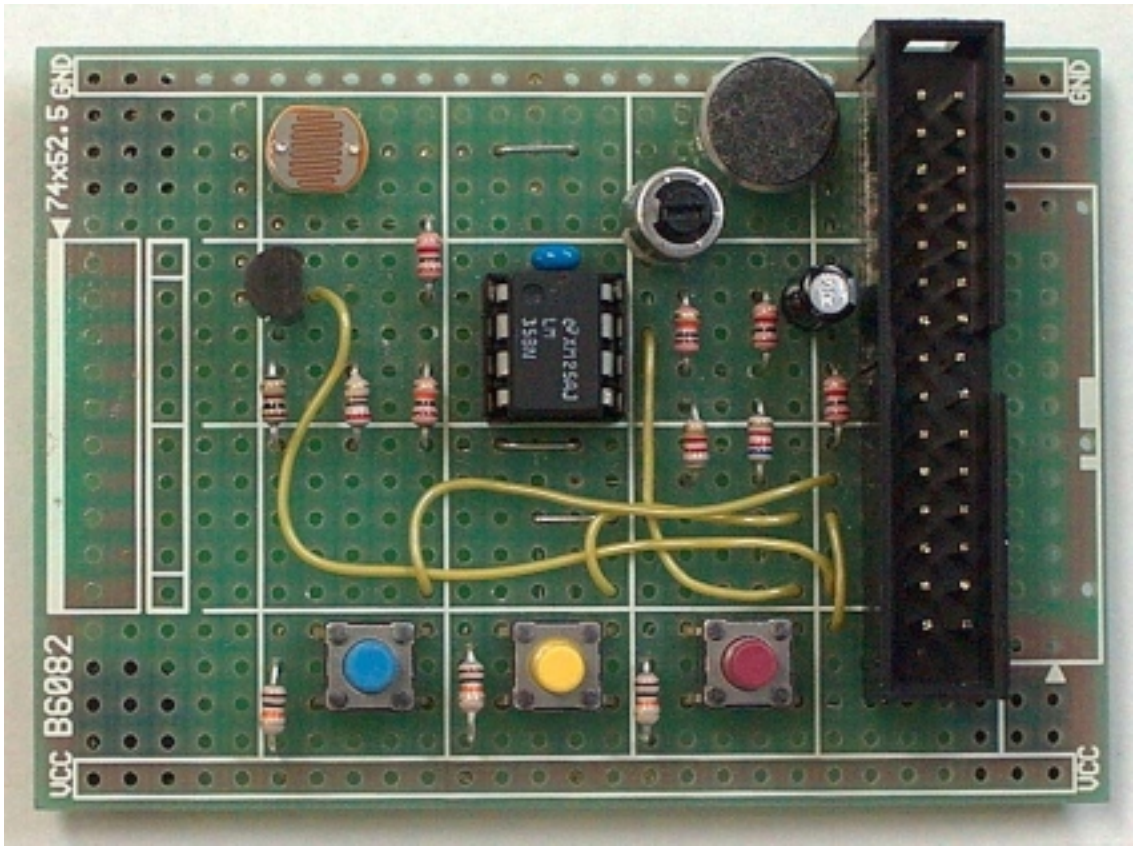
部品の確認が済みましたら、いよいよ組み立てです。次ページの回路図・実装図，その次のページの完成写真をよく見ながらハンダ付けを行って下さい。ハンダ付けによるケーブル配線は大変なので、電源や GND、交差しない信号線等はハンダ面でメッキ線や部品の余ったリード線を流用して接続します。ハンダ面のみでは配線しきれない信号線はラッピングケーブルで配線していきます。TK-3687mini との接続は前ページの写真をご覧下さい。

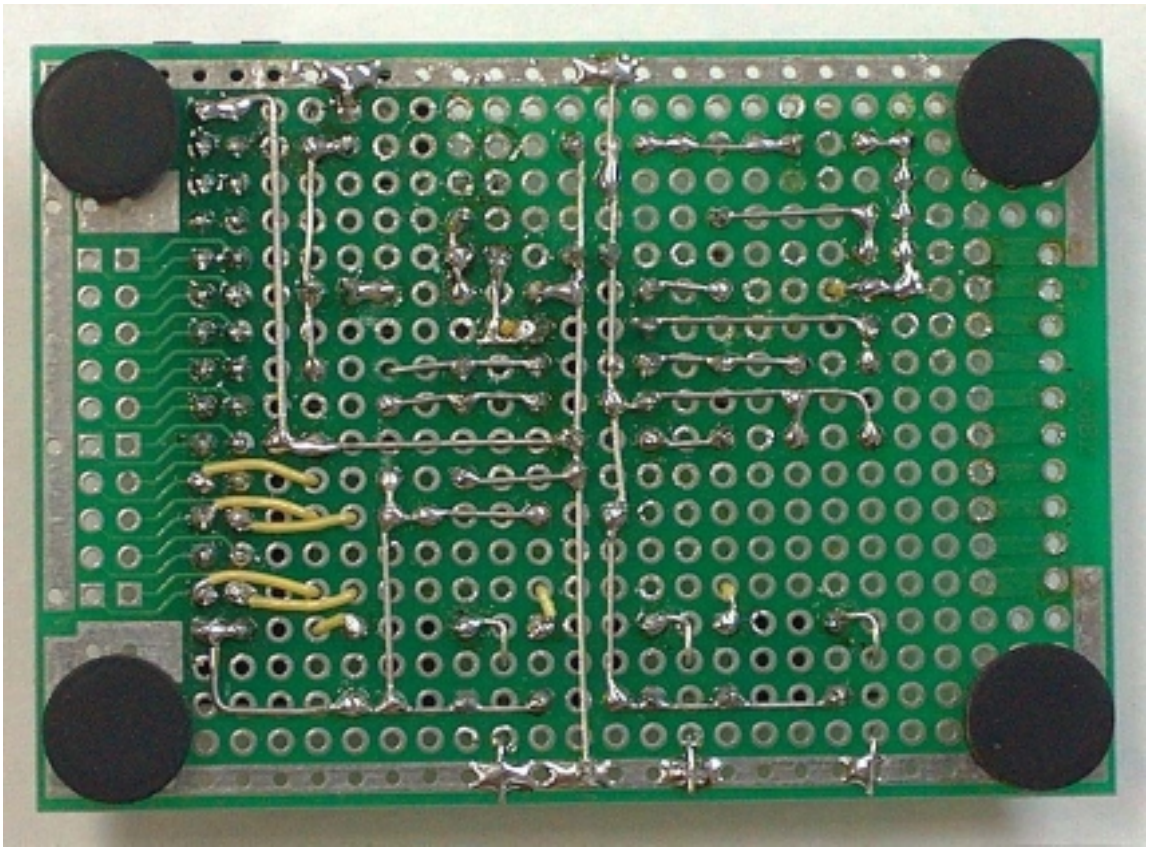
その他については TK-3687 と同じです。部品やプログラムの説明については本文をご覧下さい。

■回路図・実装図


TK-3687mini Option
A/D Converter
File : ADC(mini版).CE2
2005.06.23 Toyo-Linx.Co.,Ltd.
[Top View]
[Bottom View]
TK-3687mini(CN4)
CN4 CN1
VCC VCC PB6 PB5 PB4 PB1 PB0 GND GND
コンデンサマイクの極性について
裏から見て切り込みの入っている
パターンのリード方 "+"
パターンの外縁が黒く塗られてい
る方が "-" です。
結線について
※行番のメッキ半銅はGPIOの電源の配線に使用します。
※抵抗などの部品の足を使い半田面で結線して下さい。
※部品面のジャンパ部品の足等を使用しないでください。

■完成写真





★ご質問、お問い合わせはメール又は FAX で、、、

（株）東洋リンクス

〒102-0093　東京都千代田区平河町 1-2-2 朝日ビル

TEL: 03-3234-0559　／　FAX: 03-3234-0549

URL: http://www2.u-netsurf.ne.jp/~toyolinx

E-Mail: toyolinx@va.u-netsurf.jp

※本書の内容は将来予告無しに変更することがあります（2005 年 06 月作成）

20050628