評価ボード

アセンブラ・C 言語による
実践解説テキスト

（AHE２６１－CAT２６１　ｆｏｒ　GNU／ｇｃｃ）

初版　　２００２年　４月
第2版　２００３年　２月
第3版　２００３年　４月
第4版　２００４年　６月
第5版　２００５年　９月
第6版　２００６年　７月

Ｒｅｖ１．０６（２００６／０７／２１）

序

はじめに

この解説書は、できる限り初心者向けに解説をすることを心がけて記述しましたが、細かく書きすぎますと終わりが見えなくなってしまいます。
どの程度で記述したら良いのか非常に迷いましたが、とりあえず独断と偏見で一方的に決めさせてもらいました。
程度として、マイクロコンピュータ（マイコン）の基礎をだいたい理解しており、アセンブラおよびコンパイラは最低でも１回以上は使用したことのある方を前提にして解説を進めていきたいと思います。
また私自身も解説書を書くのは始めての経験であり、至らないところもあると思いますが皆様のご意見やご質問をもとに改訂を進め、よりよい解説書にしていきたいと思っておりますので、ご協力のほどよろしくお願いします。

エーワン㈱　技術部　長谷川正博
URL: http://www.aone.co.jp
mail: hasegawa@aone.co.jp

［変更履歴］

２００５年９月：ＤＥＦバージョン４．８０Ａより、割り込みモード０／２両対応にした為、一部説明を変更する。
２００６年７月：ＤＥＦバージョン５．７０Ａより、リセット遅延防止タイマ未使用の指定が出来るようになった為、一部説明を追加する。

［参考文献］

本書は、㈱ルネサス　テクノロジのご了解をいただき以下のマニュアルから転用しております。
Ｈ８Ｓ／２６１２シリーズ　ハードウェアマニュアル（ＡＤＪ－６０２－２４２Ａ）
㈱ルネサス　テクノロジ
マニュアルが改訂されていることもございますので、最新版はＷｅｂ上でご確認下さい。

筋書および概説

本書は、評価ボード（ＡＨＥ２６１　エーワン㈱製）と超小型マイコン（ＣＡＴ２６１　エーワン㈱製）とＨ－ｄｅｂｕｇｇｅｒ（エーワン㈱製）の教材を使用しながら、基礎から積み上げてテーマの完成を目指す実践解説書です。

マイコンのソフト開発には、全部ポーリングで済む場合と多種多用な要求に対応するため割り込みを駆使するシステムの２通り考えられます。

最近は、全部ポーリングで済むようなシステムは殆ど無く、生かさず殺さずの状態で割り込みを駆使するシステムばかりです。(個人的な感情がかなり入っています)

しかし、いきなり割り込み記述が登場しますと敷居が高く感じられてしまいますし、基礎を解説するにはチョット複雑になってしまいますので､あえて同じテーマを全ポーリングで作成した場合と、割り込みを使用して作成した場合の２通りを載せることにしました。

同じテーマを割り込み未使用／使用で作成した場合の個性がでるよう工夫をしたつもりですので違いが分かって頂けると幸いです。

最終テーマですが､評価ボードに付けたブザーを使い簡単な曲を奏でる？？装置名“メロディ♪♪”です。

かなり音痴な奴ですが、広く大きな心で我慢して聞いてやってください。

なお、本書での開発用ソフトウェアは、ＧＮＵ／ｇｃｃを使用しています。

［バージョン］

Ｃｙｇｗｉｎ	Ｖｅｒ　１．３．１２－４
Ｂｉｎｕｔｉｌｓ	Ｖｅｒ　２．１１．２
ｇｃｃ	Ｖｅｒ　２．９５．２
Ｎｅｗｌｉｂ	Ｖｅｒ　１．９．０

の環境で作成しました。

［注意事項］

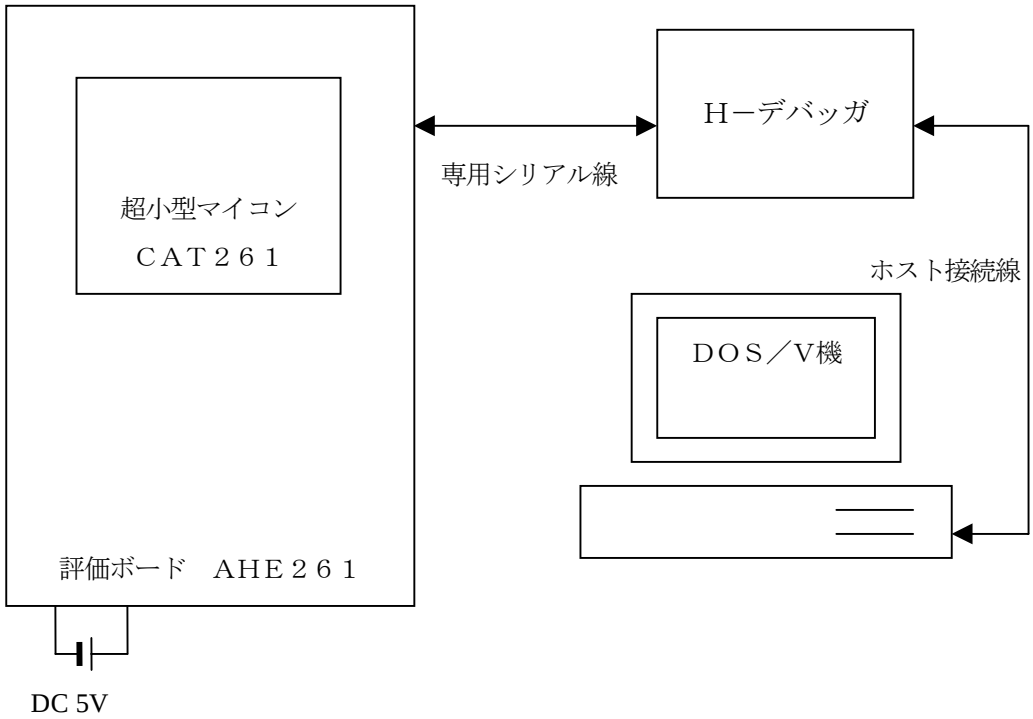
この解説書は、デバッガＩ／Ｆがブートポートタイプの**ＰＢＣ搭載ＣＰＵ**を元にした説明です。デバッガＩ／Ｆが、**Ｈ－ＵＤＩ**タイプおよび**Ｅ７／Ｅ１０**タイプとは一部相違があります。

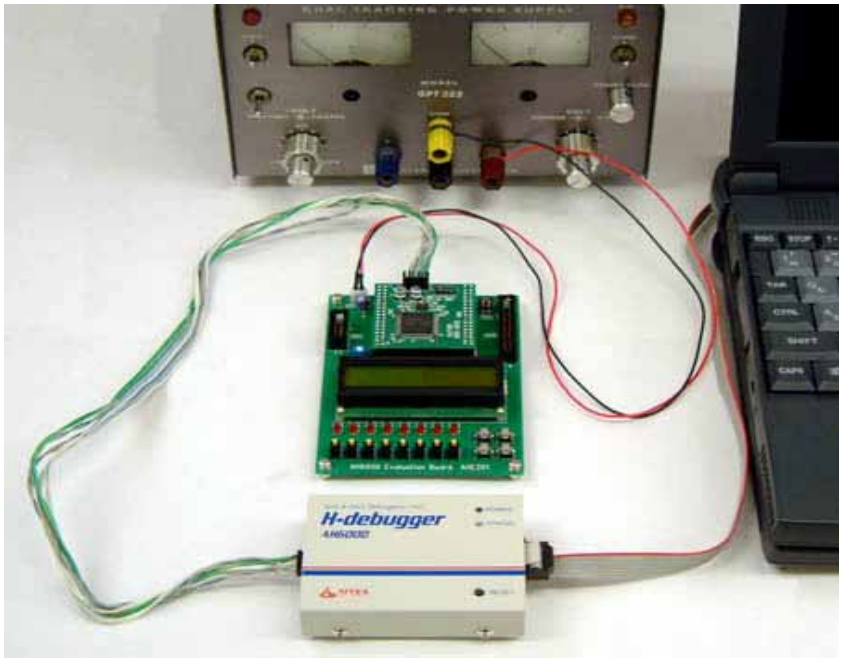
目　　次

ハード構成およびシステム構成

第１章　ハード構成

この解説書を進めるにあたり、下記ハード構成の準備をお願いします。





［接続例］

第2章　システム構成

装置名：“メロディ♪♪”のシステムブロック図およびメモリマップとＩ／Ｏ表を記述します。細かい説明は後の章にまかせ、ここではハードの全体像をつかんで下さい。

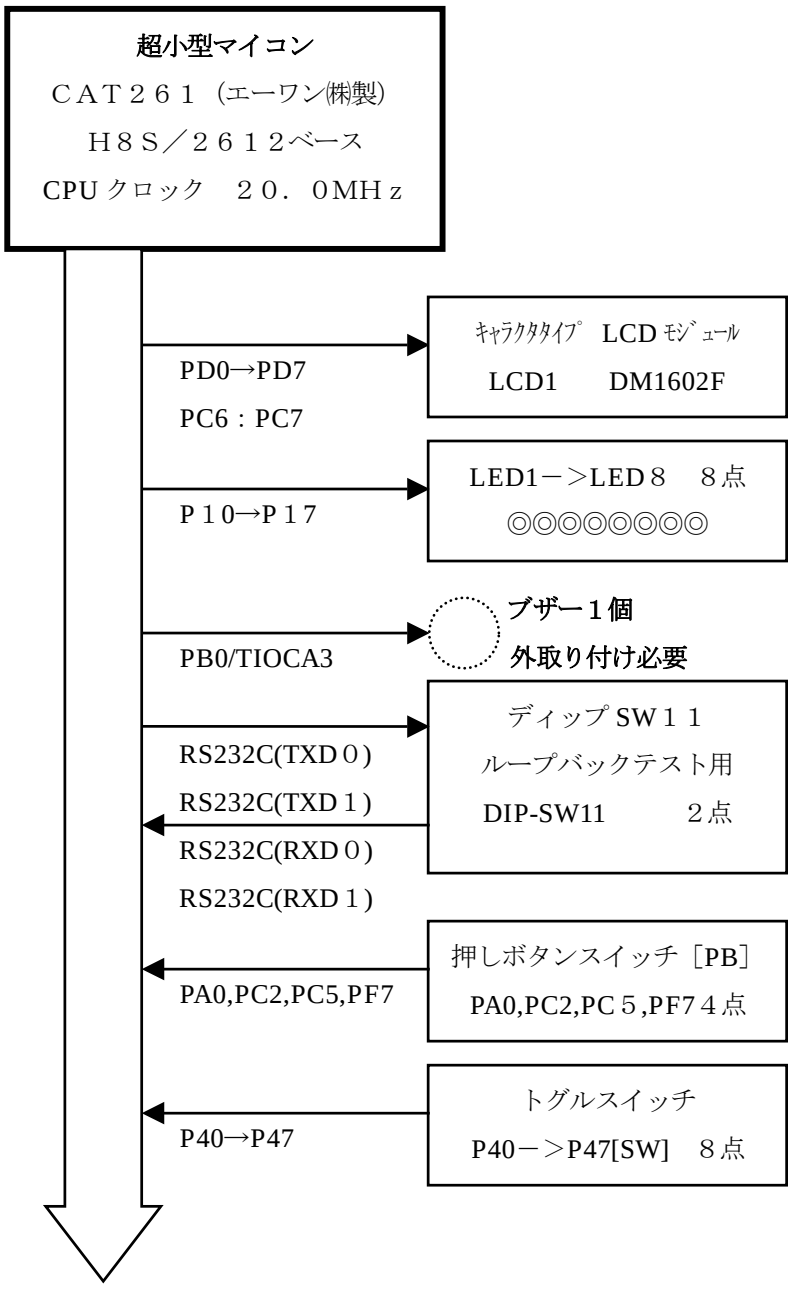
［評価ボード回路図］

“¥評価ボード¥第３部資料¥評価ボード（ＡＨＥ２６１）図面¥＊．ｐｄｆ”参照

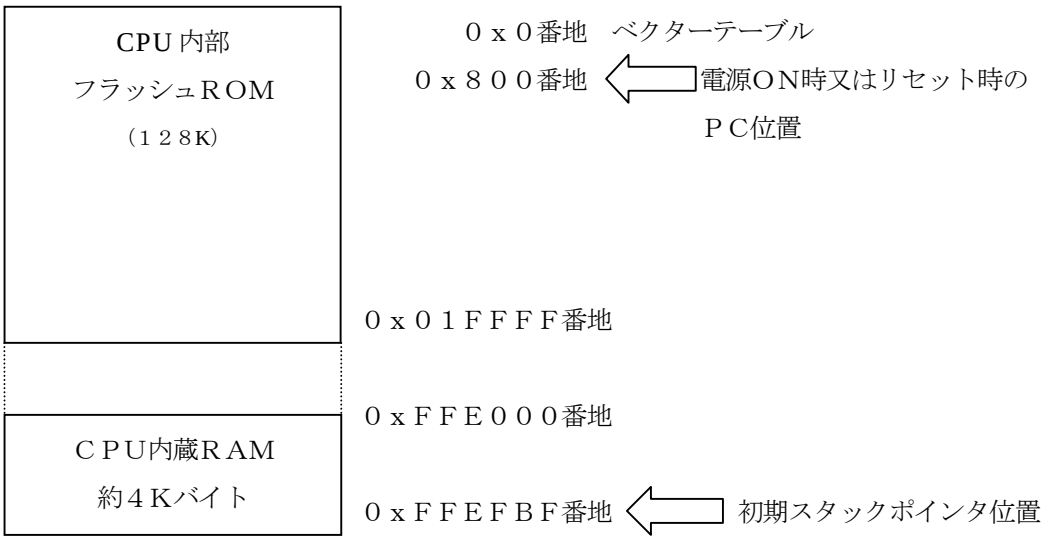
［ＣＡＴ２０４回路図］

“¥評価ボード¥第３部資料¥超小型マイコン（ＣＡＴ２６１）図面¥＊．ｐｄｆ”参照

１．システムブロック図（メロディ♪♪）



２．ＣＡＴ２６１のプログラムメモリＭＡＰ



３．評価ボードのディップ SW の設定とＩ／Ｏマップ表

ＳＷ１１の設定	ＯＮ	ＯＦＦ
ＳＷ１１－１	シリアルＩ／Ｏの TXD０,RXD１を折り返し	シリアルＩ／Ｏの TXD０,RXD１間オープン
ＳＷ１１－２	シリアルＩ／Ｏの TXD１,RXD０を折り返し	シリアルＩ／Ｏの TXD１,RXD０間オープン

ＬＣＤ関係					
ﾎﾟｰﾄｱﾄﾞﾚｽ	ﾎﾟｰﾄｼﾝﾎﾞﾙ	ピン番号	ﾋﾟﾝｼﾝﾎﾞﾙ	方向	信号名
0xffff0c	PDDR	CN1-7B	PD0	出力	D0　LCD-module
		1-7A	PD1	出力	D1
		1-6B	PD2	出力	D2
		1-6A	PD3	出力	D3
		1-5B	PD4	出力	D4
		1-5A	PD5	出力	D5
		1-4B	PD6	出力	D6
		1-4A	PD7	出力	D7
0xffff0b	PCDR	CN２-9A	PC6	出力	RS　LCD-module
		-9B	PC7	出力	E　LCD-module
					未使用
					未使用
					未使用
					未使用
					未使用
		-14A	P07		未使用

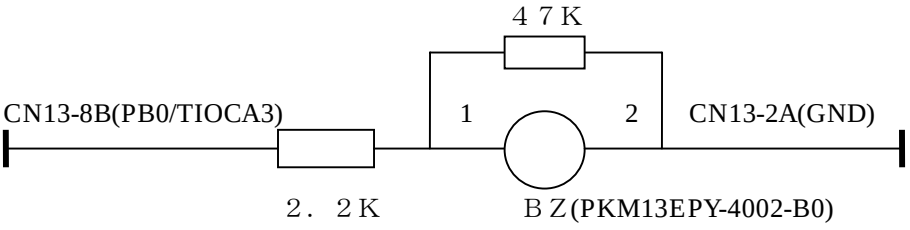
ＬＥＤ関係					
ﾎﾟｰﾄｱﾄﾞﾚｽ	ﾎﾟｰﾄｼﾝﾎﾞﾙ	ピン番号	ﾋﾟﾝｼﾝﾎﾞﾙ	方向	信号名
0xffff00	P1DR	CN1-14A	P10	出力	LED1
		1-14B	P11	出力	LED2
		1-15A	P12	出力	LED3
		1-15B	P13	出力	LED4
		1-16A	P14	出力	LED5
		1-16B	P15	出力	LED6
		1-17A	P16	出力	LED7
		1-17B	P17	出力	LED8

ブザー関係（タイマ／ＴＰＵ３　ＴＩＯＣＡ３　アウトプットコンペア出力）				
ﾎﾟｰﾄｱﾄﾞﾚｽ	ﾎﾟｰﾄｼﾝﾎﾞﾙ	ピン番号	ﾋﾟﾝｼﾝﾎﾞﾙ	仕様
0xfffe88	TGRA_3	CN2-10A	PB0/TIOCA3	ＰＷＭモード　パルス出力

押しボタンスイッチ関係					
ﾎﾟｰﾄｱﾄﾞﾚｽ	ﾎﾟｰﾄｼﾝﾎﾞﾙ	ピン番号	ﾋﾟﾝｼﾝﾎﾞﾙ	方向	信号名
0xffffb9	POATA	CN1-3B	PA0	入力	SW12　PB[PA0]
0xffffbb	PORTC	2-7A	PC2	入力	SW13　PB[PC2]
		2-8B	PC5	入力	SW14　PB[PC5]
0xffffbe	POATF	2-5B	PF7	入力	SW15　PB[PF7]

トグルスイッチ関係					
ﾎﾟｰﾄｱﾄﾞﾚｽ	ﾎﾟｰﾄｼﾝﾎﾞﾙ	ピン番号	ﾋﾟﾝｼﾝﾎﾞﾙ	方向	信号名
0xffffb3	PORT４	CN1-13B	P40	入力	SW16　SW[P40]
		1-13A	P41	入力	SW17　SW[P41]
		1-12B	P42	入力	SW18　SW[P42]
		1-12A	P43	入力	SW19　SW[P43]
		1-11B	P44	入力	SW20　SW[P44]
		1-11A	P45	入力	SW21　SW[P45]
		1-10B	P46	入力	SW22　SW[P46]
		1-10A	P47	入力	SW23　SW[P47]

外部ブザー回路図（要製作）



４．とにかく動かしてみましょう！！

１）パソコン＋Ｈ－ｄｅｂｕｇｇｅｒ＋評価ボードの接続を確認して下さい。

２）評価ボードのディップスイッチＳＷ１１－１・２両方をＯＮにする。

３）ＣＡＴ２６１基板上のディップスイッチＳＷ１－１・２・３・４をＯＮにする。

４）評価ボードの電源をオンにする。

５）ＤＥＦを立ち上げる。

６）ＤＥＦの［オプション］－［環境設定］で環境設定をする。

７）ＤＥＦの［オプション］－［ＣＰＵ設定］

ＣＰＵシリーズ名：　Ｈ８Ｓ／２６１２Ｆ－ＺＴＡＴ

水晶発振子クロック：　２０．０００００

周波数逓倍率：　ｘ１

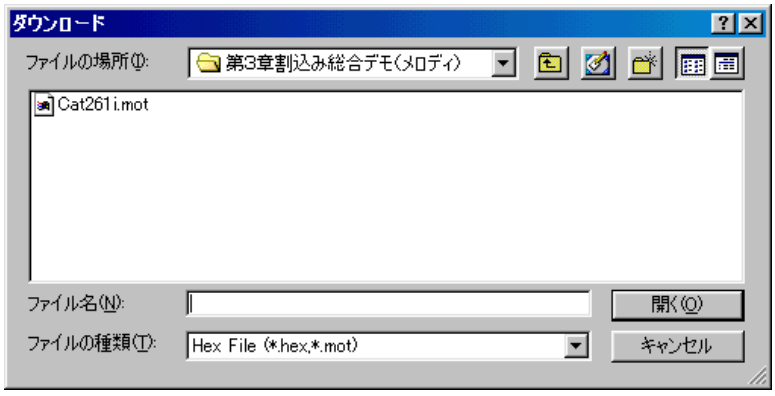
に設定してください。

８）ＤＥＦの左下Ｓｔａｒｔをクリックする。

これで、パソコンとＨ－ｄｅｂｕｇｇｅｒとＣＡＴ２６１の通信が可能になり、デバック可能となります。

９）ＤＥＦを使って、ＨＥＸファイルをダウンロードして下さい。

テーマ“メロディ♪♪”のＨＥＸファイルは、



“¥評価ボード¥第２部割込み編¥第３章割込み総合デモ（メロディ）¥Cat261i.mot“です。

（ダウンロードの方法は、ＤＥＦのＨＥＬＰを参照）

１０）ダウンロードが終了しましたら、プログラムの実行をして下さい。

（ＤＥＦのショートＰＢのＧＯをクリック）

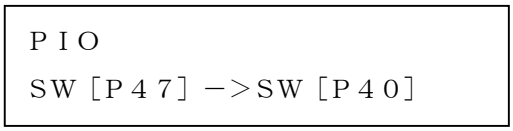
これで、ＬＥＤが何やらパラパラ表示をして、ＬＣＤに文字が表示されるはずです。

０－４－１図　オープニング表示

ここで、簡単に“メロディ♪♪”の操作手順を説明します。

ａ．まず、評価ボードにある押しボタンスイッチ（以後ＰＢと表記します）ＰＡ０を１回押してみてください。

ＰＩＯ動作モード

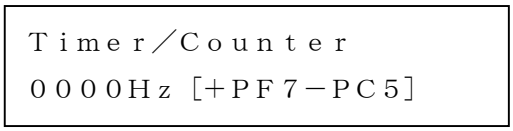


０－４－２図　ＰＩＯ表示

パラパラ表示していたＬＥＤが消灯します。
トグルスイッチ（以後ＳＷと表記します）Ｐ４７－＞Ｐ４０を上側（ＯＮ）にして見て下さい。
対角線上にＬＥＤが点灯します。
ちょっと寂しい仕様ですが、これがＰＩＯの仕様です。

ｂ．次に、ＰＢ［ＰＡ０］をもう１回押してみて下さい。

Ｔｉｍｅｒ動作モード



０－４－３図　Ｔｉｍｅｒ表示

ＰＢ［ＰＦ７］でパルス出力の周波数を１００Ｈｚ単位で上げます。
ＰＢ［ＰＣ５］でパルス出力の周波数を１００Ｈｚ単位で下げます。
ＬＣＤに現周波数が表示されます。
ブザーの音で確認ができますが、きっちりと確認したい方は、評価ボードの**ＣＮ２－１０Ａ**をオシロで確認してみて下さい。

ｃ．再度、ＰＢ［ＰＡ０］をもう一回押してみて下さい。

ＳＣＩ（ＳＩＯ）動作モード

ＳＣＩ　　８，ｎ，１［ＰＣ２］ ０９６００ｂ［＋ＰＦ７－ＰＣ５］

０－４－４図　ＳＣＩ表示

ＳＣＩ（ＲＳ２３２Ｃ）調歩同期式シリアル通信のループテストです。
表示上の“８，ｎ，１”は、８ビット、パリティなし、ストップ１ビットの意味で固定になっています。

“０９６００ｂ”は、転送ボーレートです。
ＰＢ［ＰＦ７］で転送ボーレートを上げることができます。
ＰＢ［ＰＣ５］で転送ボーレートを下げることができます。

評価ボードのＳＷ１１－１・２がＯＮになっていることを確認して下さい。（ループ接続）
ＣＡＴ２６１のＳＷ１－１・２・３・４がＯＮになっていることを確認して下さい。（ＲＳドライバへの接続）

ＰＢ［ＰＣ２］で送信開始／停止をします。
送信データは、“ＩｎｔｅｒｒｕｐｔＳＣ”の文字列を１文字ずつ空けて分けて送信しています。

ＳＣＩ０＝“Ｉ　ｔ　ｒ　ｕ　ｔ　Ｃ”　　ＳＣＩ１＝“　ｎ　ｅ　ｒ　ｐ　Ｓ　”

受信データは、ＬＣＤの１行目に表示しますので、正常受信していますと重なり合い文字として読み取れるようになっていると思います。

他に、ＳＷ１１－１をＯＦＦにした場合とか、ボーレートを上下させて、色々と操作してみて下さい。

ｄ．再度、ＰＢ［ＰＡ０］をもう一回押してみて下さい。

テーマのＭｅｌｏｄｙ（メロディ♪♪）です。

Ｍｅｌｏｄｙ ＰＣ２［Ｍ］Ｆ７［ス］Ｃ５［セ］

０－４－５図　Ｍｅｌｏｄｙ表示

ＰＢ［ＰＣ２］でモード変更出来ます。（ｏｎトグル）

ＰＣ２［Ｍ］：手動演奏モード

ＳＷ［Ｐ４０］－＞ＳＷ［Ｐ４７］をｏｎ／ｏｆｆしてみて下さい。

ＰＣ２［Ａ］：自動動演奏モード

３曲登録してあります。

（ネコフンジャッタ、イヌノオマワリサン、アマリリス）

ＰＢ［ＰＣ５］で曲を選択して下さい。

ＰＢ［ＰＦ７］で演奏の開始／停止になっています。

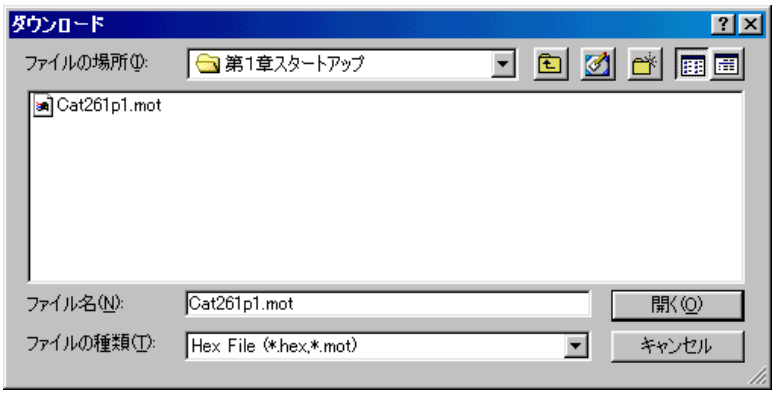
ここまでの説明で、最終目標の“メロディ♪♪”の仕様が理解していただけたでしょうか？
いよいよ、これから基礎から積み上げながら、システム名：“メロディ♪♪”の解説およびソフトウェア開発を進めていきます。

次で説明する第１部ポーリング編は、システム名：“メロディ♪♪”を割込みを一切使用せずに全ポーリング仕様で書いた場合の説明です。
そして第１部が終了しますと、第２部割込み編として割込み処理を取り入れ、よりスムーズなシステムに変更します。
これが、テーマ“メロディ♪♪”になるわけです。

第１部　ポーリング編
第1章　スタートアップ

　ここの章では、スタートアップ（電源ＯＮ時もしくはリセット時）の説明をします。
Ｈ８Ｓの場合、リセットおよび割り込みベクターは固定になっています。
そのベクターの定義ソースおよびスタートアップ部分のアセンブラーソースの説明をしたいと思います。

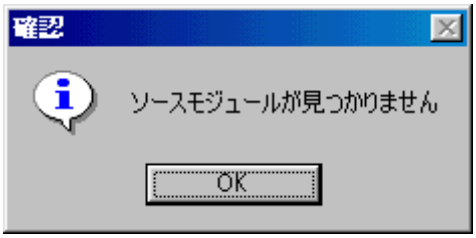
まずは、ＤＥＦのダウンロードで
¥評価ボード¥第１部ポーリング編¥第１章スタートアップ¥



にディレクトリの移動をしておいて下さい。

１．動かしてみましょう

　移動したディレクトリの中に、“Ｃａｔ２６１ｐ１．ｍｏｔ”というＨＥＸファイルがあります。
これをダウンロードしますと、このようなメッセージがでますが、ここでのプログラムはアセンブラーで作成しているため、Ｃソースがありませんとのメッセージです。



無視して“OK”をクリックして下さい。
プログラム実行してみて下さい。（ＤＥＦ画面左下のＧｏをクリック）

動作としては、なにかＬＥＤがパラパラ表示しているはずです。
このソフトは単純に２０ｍｓ毎にＬＥＤを１ビット左シフトしながら点灯させているだけのソフトです。
それでは、プログラムリストを見てみましょう！

２．プログラムリスト

１）　まずは、ベクターテーブルの設定のリスト例を示します。

file　“vectorA.asm”

```
;*************************************************************************
;*
;*  Ｈ８Ｓ　デバッカモニター用ベクターテーブル(GNU/as)
;*              TYPE:  H8S/2132.2134.2138.2612.2633F-ZTAT
;*              MODE:  Advanced
;*                                      2002/03/14 by AONE  M.Hasegawa
;*************************************************************************
            .h8300s
            .extern     _StartUp
            .extern     _non

            .section    .vector

;*********************************************************
;*  マクロ宣言       Normal = .word  Advanced = .long
;*********************************************************
            .macro  VECTOR  p
                    .long   ¥p
            .endm
;*********************************************************
;*  ベクターテーブル
;*********************************************************
_VectorTable:                       ;           (共通/213x)   (26xx)
            VECTOR      _StartUp    ;* VCT(  0) RESET       |
            VECTOR      _non        ;* VCT(  1) System 予約 1|
            VECTOR      _non        ;* VCT(  2)             2|
            VECTOR      _non        ;* VCT(  3)             3|
            VECTOR      _non        ;* VCT(  4)             4|
            VECTOR      _non        ;* VCT(  5)             5|
            VECTOR      _non        ;* VCT(  6) 直接推移    |
            VECTOR      _non        ;* VCT(  7) NMI         |
```

```
32:             VECTOR      _non            ;* VCT(  8) TRAP0       |
33:             VECTOR      _non            ;* VCT(  9) TRAP1       |
34:             VECTOR      _non            ;* VCT( 10) TRAP2       |
35:             VECTOR      _non            ;* VCT( 11) TRAP3       |
36:             VECTOR      _non            ;* VCT( 12) System 予約 1|
37:             VECTOR      _non            ;* VCT( 13)            2|
38:             VECTOR      _non            ;* VCT( 14)            3|
39:             VECTOR      _non            ;* VCT( 15)            4|
40:             VECTOR      _non            ;* VCT( 16) IRQ0        |
41:             VECTOR      _non            ;* VCT( 17) IRQ1        |
42:             VECTOR      _non            ;* VCT( 18) IRQ2        |
43:             VECTOR      _non            ;* VCT( 19) IRQ3        |
44:             VECTOR      _non            ;* VCT( 20) IRQ4        |
45:             VECTOR      _non            ;* VCT( 21) IRQ5        |
46:             VECTOR      _non            ;* VCT( 22) IRQ6        |
47:             VECTOR      _non            ;* VCT( 23) IRQ7        |
48:             VECTOR      _non            ;* VCT( 24) DTC   END   |
49:             VECTOR      _non            ;* VCT( 25) WDG   WOVI0 |
50:             VECTOR      _non            ;* VCT( 26) WOVI1       |
51:             VECTOR      _non            ;* VCT( 27) PCB         |
52:             VECTOR      _non            ;* VCT( 28) A/D   ADI   |
53:             VECTOR      _non            ;* VCT( 29) ﾘｻﾞｰﾌﾞ       |
54:             VECTOR      _non            ;* VCT( 30) ﾘｻﾞｰﾌﾞ       |
55:             VECTOR      _non            ;* VCT( 31) ﾘｻﾞｰﾌﾞ       |
56:             VECTOR      _non            ;* VCT( 32) ﾘｻﾞｰﾌﾞ       |TPU_0 TGIA_0
57:             VECTOR      _non            ;* VCT( 33) ﾘｻﾞｰﾌﾞ       |      TGIB_0
58:             VECTOR      _non            ;* VCT( 34) ﾘｻﾞｰﾌﾞ       |      TGIC_0
59:             VECTOR      _non            ;* VCT( 35) ﾘｻﾞｰﾌﾞ       |      TGID_0
60:             VECTOR      _non            ;* VCT( 36) ﾘｻﾞｰﾌﾞ       |      TCIV_0
61:             VECTOR      _non            ;* VCT( 37) ﾘｻﾞｰﾌﾞ       |
62:             VECTOR      _non            ;* VCT( 38) ﾘｻﾞｰﾌﾞ       |
63:             VECTOR      _non            ;* VCT( 39) ﾘｻﾞｰﾌﾞ       |
64:             VECTOR      _non            ;* VCT( 40) ﾘｻﾞｰﾌﾞ       |TPU_1 TGIA_1
65:             VECTOR      _non            ;* VCT( 41) ﾘｻﾞｰﾌﾞ       |      TGIB_1
66:             VECTOR      _non            ;* VCT( 42) ﾘｻﾞｰﾌﾞ       |      TGIV_1
67:             VECTOR      _non            ;* VCT( 43) ﾘｻﾞｰﾌﾞ       |      TGIU_1
```

```
68:             VECTOR      _non            ;* VCT( 44) ﾘｻﾞｰﾌﾞ       |TPU_2 TGIA_2
69:             VECTOR      _non            ;* VCT( 45) ﾘｻﾞｰﾌﾞ       |       TGIB_2
70:             VECTOR      _non            ;* VCT( 46) ﾘｻﾞｰﾌﾞ       |       TGIV_2
71:             VECTOR      _non            ;* VCT( 47) ﾘｻﾞｰﾌﾞ       |       TGIU_2
72:             VECTOR      _non            ;* VCT( 48) FRT   ICIA  |TPU_3 TGIA_3
73:             VECTOR      _non            ;* VCT( 49)        ICIB  |       TGIB_3
74:             VECTOR      _non            ;* VCT( 50)        ICIC  |       TGIC_3
75:             VECTOR      _non            ;* VCT( 51)        ICID  |       TGID_3
76:             VECTOR      _non            ;* VCT( 52)        OCIA  |       TCIV_0
77:             VECTOR      _non            ;* VCT( 53)        OCIB  |
78:             VECTOR      _non            ;* VCT( 54)        FOVI  |
79:             VECTOR      _non            ;* VCT( 55) ﾘｻﾞｰﾌﾞ       |
80:             VECTOR      _non            ;* VCT( 56) ﾘｻﾞｰﾌﾞ       |TPU_4 TGIA_4
81:             VECTOR      _non            ;* VCT( 57) ﾘｻﾞｰﾌﾞ       |       TGIB_4
82:             VECTOR      _non            ;* VCT( 58) ﾘｻﾞｰﾌﾞ       |       TGIV_4
83:             VECTOR      _non            ;* VCT( 59) ﾘｻﾞｰﾌﾞ       |       TGIU_4
84:             VECTOR      _non            ;* VCT( 60) ﾘｻﾞｰﾌﾞ       |TPU_5 TGIA_5
85:             VECTOR      _non            ;* VCT( 61) ﾘｻﾞｰﾌﾞ       |       TGIB_5
86:             VECTOR      _non            ;* VCT( 62) ﾘｻﾞｰﾌﾞ       |       TGIV_5
87:             VECTOR      _non            ;* VCT( 63) ﾘｻﾞｰﾌﾞ       |       TGIU_5
88:             VECTOR      _non            ;* VCT( 64) TMR0  CMIA0 |
89:             VECTOR      _non            ;* VCT( 65)       CMIB0 |
90:             VECTOR      _non            ;* VCT( 66)       OVI0  |
91:             VECTOR      _non            ;* VCT( 67) ﾘｻﾞｰﾌﾞ       |
92:             VECTOR      _non            ;* VCT( 68) TMR1  CMIA1 |
93:             VECTOR      _non            ;* VCT( 69)       CMIB1 |
94:             VECTOR      _non            ;* VCT( 70)       OVI1  |
95:             VECTOR      _non            ;* VCT( 71) ﾘｻﾞｰﾌﾞ       |
96:             VECTOR      _non            ;* VCT( 72) TMRXY CMIAY |
97:             VECTOR      _non            ;* VCT( 73)       CMIBY |
98:             VECTOR      _non            ;* VCT( 74)       OVIY  |
99:             VECTOR      _non            ;* VCT( 75)       ICIX  |
100:            VECTOR      _non            ;* VCT( 76)       IBF1  |
101:            VECTOR      _non            ;* VCT( 77)       IBF2  |
102:            VECTOR      _non            ;* VCT( 78) ﾘｻﾞｰﾌﾞ       |
103:            VECTOR      _non            ;* VCT( 79) ﾘｻﾞｰﾌﾞ       |
```

```
104:            VECTOR      _non            ;* VCT( 80) SCI0  ERI0  |
105:            VECTOR      _non            ;* VCT( 81)       RXI0  |
106:            VECTOR      _non            ;* VCT( 82)       TXI0  |
107:            VECTOR      _non            ;* VCT( 83)       TEI0  |
108:            VECTOR      _non            ;* VCT( 84) SCI1  ERI1  |
109:            VECTOR      _non            ;* VCT( 85)       RXI1  |
110:            VECTOR      _non            ;* VCT( 86)       TXI1  |
111:            VECTOR      _non            ;* VCT( 87)       TEI1  |
112:            VECTOR      _non            ;* VCT( 88) SCI2  ERI2  |
113:            VECTOR      _non            ;* VCT( 89)       RXI2  |
114:            VECTOR      _non            ;* VCT( 90)       TXI2  |
115:            VECTOR      _non            ;* VCT( 91)       TEI2  |
116:            VECTOR      _non            ;* VCT( 92) IIC0  IICI0 |TMR2  CMIA2
117:            VECTOR      _non            ;* VCT( 93)       DDCSW1|      CMIB2
118:            VECTOR      _non            ;* VCT( 94) IIC1  IICI1 |      OVI2
119:            VECTOR      _non            ;* VCT( 95) ﾘｻﾞｰﾌﾞ      |
120:            VECTOR      _non            ;* VCT( 96) ﾘｻﾞｰﾌﾞ      |TMR3  CMIA3
121:            VECTOR      _non            ;* VCT( 97) ﾘｻﾞｰﾌﾞ      |      CMIB3
122:            VECTOR      _non            ;* VCT( 98) ﾘｻﾞｰﾌﾞ      |      OVI3
123:            VECTOR      _non            ;* VCT( 99) ﾘｻﾞｰﾌﾞ      |
124:            VECTOR      _non            ;* VCT(100) ﾘｻﾞｰﾌﾞ      |IIC0  IICI0
125:            VECTOR      _non            ;* VCT(101) ﾘｻﾞｰﾌﾞ      |      DDCSW1
126:            VECTOR      _non            ;* VCT(102) ﾘｻﾞｰﾌﾞ      |IIC1  IICI1
127:            VECTOR      _non            ;* VCT(103) ﾘｻﾞｰﾌﾞ      |
128:            VECTOR      _non            ;* VCT(104) ﾘｻﾞｰﾌﾞ      |HCAN  ERS0,OVR0
129:            VECTOR      _non            ;* VCT(105) ﾘｻﾞｰﾌﾞ      |      RM0
130:            VECTOR      _non            ;* VCT(106) ﾘｻﾞｰﾌﾞ      |      RM1
131:            VECTOR      _non            ;* VCT(107) ﾘｻﾞｰﾌﾞ      |      SLE0
132:            VECTOR      _non            ;* VCT(108) ﾘｻﾞｰﾌﾞ      |MMT   TGIMN
133:            VECTOR      _non            ;* VCT(109) ﾘｻﾞｰﾌﾞ      |      TGINN
134:            VECTOR      _non            ;* VCT(110) ﾘｻﾞｰﾌﾞ      |      POEIN
135:            VECTOR      _non            ;* VCT(111) ﾘｻﾞｰﾌﾞ      |
136:            VECTOR      _non            ;* VCT(112) ﾘｻﾞｰﾌﾞ      |
137:            VECTOR      _non            ;* VCT(113) ﾘｻﾞｰﾌﾞ      |
138:            VECTOR      _non            ;* VCT(114) ﾘｻﾞｰﾌﾞ      |
139:            VECTOR      _non            ;* VCT(115) ﾘｻﾞｰﾌﾞ      |
```

```
140:                VECTOR      _non            ;* VCT(116) ﾘｻﾞｰﾌﾞ       |
141:                VECTOR      _non            ;* VCT(117) ﾘｻﾞｰﾌﾞ       |
142:                VECTOR      _non            ;* VCT(118) ﾘｻﾞｰﾌﾞ       |
143:                VECTOR      _non            ;* VCT(119) ﾘｻﾞｰﾌﾞ       |
144:                VECTOR      _non            ;* VCT(120) ﾘｻﾞｰﾌﾞ       |SCI_3 ERI2
145:                VECTOR      _non            ;* VCT(121) ﾘｻﾞｰﾌﾞ       |      RXI2
146:                VECTOR      _non            ;* VCT(122) ﾘｻﾞｰﾌﾞ       |      TXI2
147:                VECTOR      _non            ;* VCT(123) ﾘｻﾞｰﾌﾞ       |      TEI2
148:                VECTOR      _non            ;* VCT(124) ﾘｻﾞｰﾌﾞ       |SCI_3 ERI2
149:                VECTOR      _non            ;* VCT(125) ﾘｻﾞｰﾌﾞ       |      RXI2
150:                VECTOR      _non            ;* VCT(126) ﾘｻﾞｰﾌﾞ       |      TXI2
151:                VECTOR      _non            ;* VCT(127) ﾘｻﾞｰﾌﾞ       |      TEI2
152:                .end
```

［リストの説明］

１～7行：

コメントです。

8行：

CPU タイプの擬似命令です。

Ｈ８Ｓの場合は、“．ｈ８３００ｓ”　Ｈ８／３００Ｈの場合は、“．ｈ８３００”と置きます。

９～１０行：

外部参照の宣言です。

１２行：

ベクターテーブルのセクション名を、“．ｖｅｃｔｏｒ”と定義します。

１７～１９行：

ベクターテーブルに置くアドレス値を、ノーマルモードは２バイト“．ｗｏｒｄ”でアドバンスドは４バイト“．ｌｏｎｇ”になりますので、ここのマクロを変更することによりアドレス長を変えられるよう工夫がしてあります。

１３～１５１行：

ベクターテーブルです。今回は１３行目のリセットベクターのみにアドレスを置いています。

１５２行：

アセンブラファイルの最後には、この擬似命令“．ｅｎｄ”を置いて下さい。

２）　つぎに、スタートアップのリスト例を示します。

file　“StartupA.asm”

```
;/*************************************************************************/
;/* ＜サンプル＞　ポーリング                                              */
;/*                                                                       */
;/*      ＜MOD＞      StartupA.asm                                        */
;/*      ＜役割＞     main                                                */
;/*      ＜タブ＞     ４タブ編集       GNU/as                             */
;/*                   スタートアップ（CAT261-H8S/2612）                   */
;/*************************************************************************/
            .h8300s
            .section    .text
            .global     _StartUp,_non

            ;--------   I/O         ------
            .EQU        SYSCR,      0xfffde5        ; ｼｽﾃﾑｺﾝﾄﾛｰﾙ
            .EQU        P1DDR,      0x0FFFE30       ; Moniter  Port DIR
            .EQU        P1DR,       0x0FFFF00       ; Moniter  Port Out
;/*************************************************************************/
;/*          スタートアップ                                               */
;/*************************************************************************/
_StartUp:
            mov.l       #0xffefbe,sp    ;* ｽﾀｯｸﾎﾟｲﾝﾀ 設定                 */
            ldc.b       #0x6,exr        ;* disable() PBC の割込の為        */
            jmp         _main           ;* Main

_non:                                   ;* 未使用割り込み                 */
            rte
;/*************************************************************************/
;/*          メイン                                                       */
;/*************************************************************************/
_main:
            bset        #5,@SYSCR       ;* ｼｽﾃﾑｺﾝﾄﾛｰﾗ 割込ﾓｰﾄﾞ 2           */
            bclr        #4,@SYSCR       ;*              〃                 */
            bclr        #3,@SYSCR       ;*          NMI 立下りエッジ       */
```

```
34:             bset        #1,@SYSCR       ;*              RAME 有効               */
35:
36:             mov.b       #0xff,r0l       ;* Moniter      Port All Out            */
37:             mov.b       r0l,@P1DDR      ;*                    〃                */
38:             mov.b       #1,r1l          ;* LED          表示パターン            */
39: loop:
40:             mov.b       r1l,r0l         ;* LED          点灯                    */
41:             not.b       r0l             ;*              0=点灯のため            */
42:             mov.b       r0l,@P1DR       ;*              出力 (LED 点灯)         */
43:             mov.b       #20,r0l         ;* 20msWait                             */
44: wait20ms:
45:             bsr         Wait1ms
46:             dec.b       r0l
47:             bne         wait20ms
48:             rotl.b      r1l             ;* LED          << 1                    */
49:             bra         loop
50: ;/*************************************************************************/
51: ;/*     Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (20.0000MHz)  Non Wait                    */
52: ;/*************************************************************************/
53: Wait1ms:
54:             push.w      r0
55:             mov.w       #5000,r0
56: wait:
57:             dec.w       #1,r0           ;* 1 clock
58:             bne         wait:16         ;* 3 clock
59:             pop.w       r0
60:             rts
61:             .end
```

［リストの説明］

１０行：

プログラムであることを示すセクション宣言です。“．ｔｅｘｔ”は一般的ですので、このセクション名を使うことをお勧めします。

なお、Ｃ言語で記述した場合も、このセクション名になります。

１１行：

外部参照が必要なシンボルは、このグローバル宣言をする必要があります。

“_ＳｔａｒｔＵＰ”と“_ｎｏｎ”は、“ｖｅｃｔｏｒＡ．ａｓｍ”で使用しています。

１１行：

Ｉ／Ｏレジスタアドレスをシンボル化するための宣言です。

シンボル名の最後の“,”（カンマ）をお忘れなく！

２０行：

電源ＯＮ時およびリセット時に**プログラムはココから実行を始める**ようにベクターテーブルに設定してあります。

２１行：

電源ＯＮ時およびリセット時にまず実行させる命令は、このスタックポインタの設定です。

定石ですので必ず守って下さい。

なお、理由はＣＰＵハードウェアマニュアルに記載されています。

２２行：

Ｈ８Ｓ／２６１２のＰＣブレーク例外処理の割り込みプライオリティが“7”になっていますので、ＰＣブレーク例外処理を有効にさせるために、この命令を置いています。

他の割り込み例外処理のプライオリティを“6”以下にする必要があります。

２３行：

メインプログラムへのジャンプエントリです。

２５～２６行：

未使用割り込みのベクター値を置くためのプログラムです。

実際には、このプログラムが実行されることは通常状態である限りありえません。

ここまでがポーリングで済む場合の、世でいう“**スタートアップ**”になるわけです。

どうです簡単でしょう！！

３０行目以降は、このサンプルソフトが動作しているかを目で確かめるために用意したソフトです。

ＬＥＤ点灯部分等は、後章ＰＩＯ編で説明しますので、ここでは省略します。

３．保守ツール（Ｍａｋｅｆｉｌｅ）

コンパイル・アセンブリ・リンケージエディタを管理／制御する保守ツールは、ｍａｋｅ．ｅｘｅ）を使いました。

“Ｍａｋｅｆｉｌｅ”にこのプロジェクト（テーマ）の定義をしておけば、“ｍａｋｅ”とコマンドを打つだけで、自動で修正したソースだけを探して、コンパイル・アセンブリ・リンクを実行してくれる便利なツールです。

Ｍａｋｅｆｉｌｅの仕様を、簡単ではありますが、ここで説明をしておきます。

１）この章のサンプルで使用したＭａｋｅｆｉｌｅの中身

file “Makefile”

```
1: AS          = h8300-hms-coff-as
2: CC          = h8300-hms-coff-gcc
3: LD          = h8300-hms-coff-ld
4: OBJCOPY = h8300-hms-coff-objcopy
5: SYM         = coffext
6:
7: TARGET      = Cat261p1.mot
8: OBJS        = vectorA.o  StartupA.o
9: LOCATE      = h8s261x.x
10:
11: LIBDIR     = /usr/local/h8/h8300-hms-coff/lib/h8300s/
12: INCLUDE = -I/usr/local/h8/h8300-hms-coff/include/
13: LIBC       = $(LIBDIR)libm.a $(LIBDIR)libc.a $(LIBDIR)libgcc.a
14: CCFLAGS = -S -ms -g -nostartfiles
15: # LDFLAGS = -v -t
16:
17: .SUFFIXES: .coff .o .c .s .asm .h .mot
18:
19: $(TARGET): $(OBJS)
20:     $(LD)  $(LDFLAGS) -o $*.coff -Map $*.map -T $(LOCATE) $(OBJS) $(LIBC)
21:     $(OBJCOPY) -O srec $*.coff $*.mot
22:     $(SYM) $*.coff
23:
24: .asm.o:
25:     $(AS) -o $*.o $*.asm
```

```
26:
27: .c.o:
28:     $(CC) $(CCFLAGS) $(INCLUDE) $*.c
29:     $(AS) -o $*.o $*.s
30:
```

［Ｍａｋｅｆｉｌｅの説明］

1行：

環境変数“ＡＳ”に、アセブラコマンド名を登録しています。

2行：

環境変数“ＣＣ”に、Ｃコンパイラコマンド名を登録しています。

3行：

環境変数“ＬＤ”に、リンクエディタコマンド名を登録しています。

4行：

環境変数“ＯＢＪＣＯＰＹ”に、ＨＥＸファイル作成コマンド名を登録しています。

5行：

環境変数“ＳＹＭ”に、デバッガ用シンボルコンバータコマンド名を登録しています。

7行：

環境変数“ＴＡＲＧＥＴ”に、プロジェクト（ＨＥＸファイル）名を登録しています。

8行：

環境変数“ＯＢＪ”に、プロジェクトを作成するにあたり、必要なプログラムモジュールのオブジェクト（コンパイル／アセンブリが作成する）名を登録しています。

ＧＮＵ／ｇｃｃでは、“＊．ｏ”になります。

9行：

環境変数“ＬＯＣＡＴＥ”に、番地の割り振り（ロケート）に必要なセグメント別番地情報を定義したスクリプトファイル名を登録しています。

このファイルの中については、後で説明します。

１１行：

環境変数“ＬＩＢＤＩＲ”に、コンパイラが使用するライブラリのディレクトリ位置を登録しています。

ＧＮＵコマンドを走らせる環境に“ｃｙｇｗｉｎ”を使用していますので、ｌｉｎｕｘ風になるわけです。

ｃｙｇｗｉｎのルールとして、￥ｃｙｇｗｉｎの中をｒｏｏｔ（“／”）としていますので、このような設定になります。

対比するために、Ｗｉｎｄｏｗｓ風に書きますと、

“￥ｃｙｇｗｉｎ￥ｕｓｒ￥ｌｏｃａｌ￥ｈ８￥ｈ８３００－ｈｍｓ－ｃｏｆｆ￥ｌｉｂ￥ｈ８３００ｓ￥”

となります。

１２行：

環境変数“ＩＮＣＬＵＤＥ”に、ｇｃｃの標準ライブラリ用ヘッダーファイルのディレクトリ場所を登録しています。

１３行：

環境変数“ＬＩＢＣ”に、ｇｃｃの標準ライブラリ名を登録しています。

１４行：

環境変数“ＣＣＦＬＡＧＳ”に、ｇｃｃコンパイラスイッチを登録しています。

このコンパイラスイッチは、最低限必要なスイッチです。他に必要な場合は、ｇｃｃのマニュアルをＷｅｂから入手してお使い下さい。

１５行：

リンカスイッチで、詳細情報を出力させるスイッチです。

チョットうるさいのでコメントアウトしてあります。

１７行：

“.SUFFIXES”は、中で使用する拡張子の登録です。

注意としては、小文字で登録した場合は、小文字で記述して下さい。

１９行：

“$(TARGET) : $(OBJS)”は、前に登録した（プロジェクト名）と（オブジェクト名リスト）の作成時期の関係を調べます。

（プロジェクト名）のほうが古い（昔にできた）場合、次行のリンクコマンド等を実行させるための判定文です。

“このへんが自動で判断してくれるのでヒジョウに便利です。”

２０行：

リンクコマンド行です。注意事項として、行の先頭は、<TAB>送りをしてください。

２１行：

ＨＥＸファイル作成コマンド行です。注意事項として、行の先頭は、<TAB>送りをしてください。

２２行：

デバッガ用シンボルコンバタコマンド行です。注意事項として、行の先頭は、<TAB>送りをしてください。

２４行：

“.asm.o:”は、Ａｓｍソースファイル“．ａｓｍ”とオブジェクトファイル“．ｏ”の作成時期の関係を調べます。

この場合も、オブジェクトが古い（昔にできた）場合、次行のアセンブリコマンドを実行します。

２７行：

“.c.o:”は、Ｃソースファイル“．ｃ”とオブジェクトファイル“．ｓｏｆ”の作成時期の関係を調べます。

この場合は、オブジェクトが古い（昔にできた）場合、次行のコンパイルコマンドを実行します。

今後、新規プロジェクトを開発することになり、Ｍａｋｅｆｉｌｅを作成する必要になった場合、このＭａｋｅｆｉｌｅの７、８、９行の変更で対応することが出来ると思いますので参考にして頂ければ幸いです。

2）LD コマンド用スクリプトファイル

セクション事のグループ指定および、アドレスを指定する定義ファイルです。

このファイルは、ＬＤコマンド（リンクエディタ）の時に使用します。

file “h8s261x.x”

```
 1: OUTPUT_FORMAT(″coff-h8300″)
 2: OUTPUT_ARCH(h8300S)
 3:
 4: SECTIONS
 5: {
 6:    .vect    0x0       : { *(.vector) ; }
 7:    .rom     0x800     : { *(.text) *(.rodata) *(.string) _etext = . ; }
 8:    .const   .         : { *(.data) _edata = . ;}
 9:    .bss     0xffe000  : { *(.bss) *(COMMON) *(.stack) _end = . ; }
10: }
11:
```

［説明］

1行：

ＬＤコマンドでの出力オブジェクトタイプの指定です。

Ｈ８／３００Ｈ ――――＞ OUTPUT_FORMAT("coff-h8300")

Ｈ８Ｓシリーズ ――――＞ OUTPUT_FORMAT("coff-h8300")

ＳＨ－２ ――――――＞ OUTPUT_FORMAT("coff-sh")

2行：

ＣＰＵアーキテクチャの指定です。

Ｈ８／３００Ｈ ――――＞ OUTPUT_ARCH(h8300H)

Ｈ８Ｓシリーズ ――――＞ OUTPUT_ARCH(h8300Ｓ)

ＳＨ－２ ――――――＞ OUTPUT_ARCH(sh)

4～5行：

セクション指定の始まりを意味します。

6行：

“vectorA.asm”で宣言したセクション名のアドレス指定です。必ずゼロ（０）にして下さい。

7行：

プログラムの開始アドレスの指定です。必ず（０ｘ８００）にして下さい。

この指定により、モニタエリアの０ｘ２００～０ｘ７ｆｆが確保されることになります。

＊（．ｔｅｘｔ） ＝ プログラムセクション名

＊(．ｒｏｄａｔａ)　＝　ｃｏｎｓｔ宣言をしたセクション名

＊(．ｓｔｒｉｎｇ)　＝　プログラム中に文字列記述“ｘｘｘｘ”したセクション名

＿ｅｔｅｘｔ　　＝　．ｒｏｍグループの最後を示します

8行：

変更可能な初期化変数指定をした場合のセクションになります。

例　ｃｈａｒ　ａ＝１０

ただし、ＲＯＭ化システムの場合は、使用しないで下さい。

＊(．ｄａｔａ)　＝　変更可能な初期化変数宣言をしたセクション名

＿ｅｄａｔａ　　＝　ＲＯＭエリア確保の最終を示します。

9行：

ＲＡＭエリアの開始アドレスを指定します。

＊(．ｂｓｓ)　　＝　ｓｔａｔｉｃ宣言をしたセクション名

＊(ＣＯＭＭＯＮ)　＝　グロバール変数のセクション名

＊(．ｓｔａｃｋ)　＝　スタックセクション名

＿ｅｎｄ　　＝　ＲＡＭエリア確保の最終を示します

１０行：

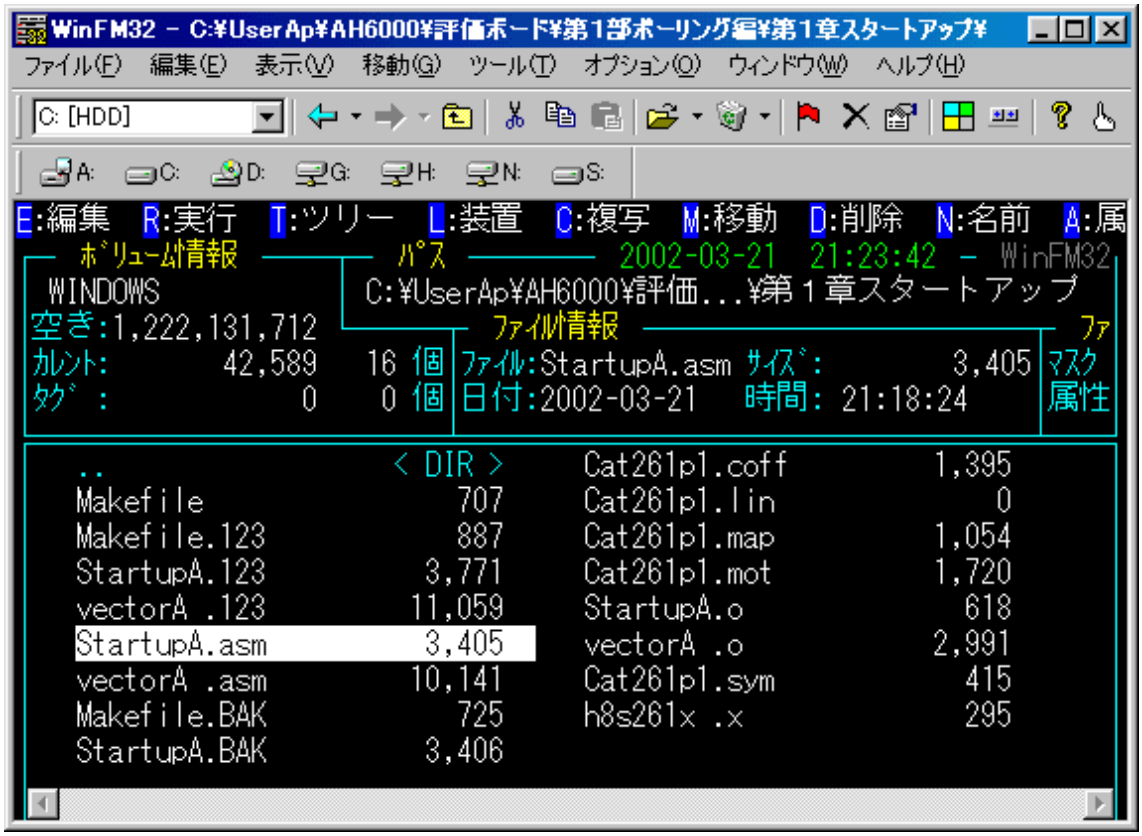
セクション指定の最終を示します。

３）ｍａｋｅの実行

一度、ｍａｋｅの便利さを実感して見ましょう。

このサンプルでＬＥＤを２０ｍｓ毎にシフト表示しているのを１００ｍｓ毎に変更してみましょう。

まずは、何かのファイラーで“¥評価ボード¥第１部ポーリング編¥第１章スタートアップ¥”のディレクトリに移動します。



“ＳｔａｒｔｕｐＡ．ａｓｍ”をエディタで開き

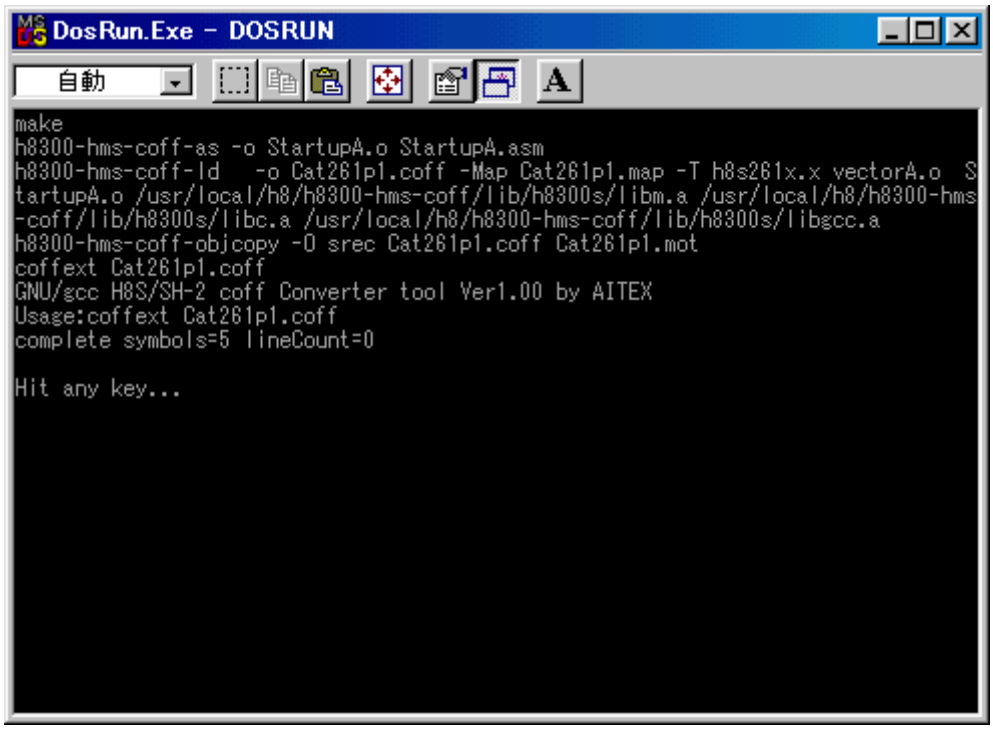
```
43:             mov.b       #20,r0l         ;* 20msWait                    */
44: wait20ms:
45:             bsr         Wait1ms
```

４３行目の“２０”を“１００”に変更してみてください。

```
43:             mov.b       #100,r0l        ;* 100msWait                   */
```

変更後、保存終了して下さい。

保存終了が済みましたら、“ｍａｋｅ”を実行してみて下さい。



このような実行結果が表示されるはずです。

Ｍａｋｅｆｉｌｅの機能により、プロジェクト名：Ｃａｔ２０４ｐ１．ｍｏｔが新しく作成されました。

評価ボードにダウンロードして、実行してみて下さい。

どうです！！ＬＥＤのシフト表示スピードが遅くなりましたので、目で確認できるようになったはずです。

プログラムデバック作業は、簡単に言えばこの繰り返しです。

思ったように動かない場合は、デバッガでブレークをあてたり、メモリの内容を見たり、レジスタの内容を見たりして、間違いを探し、見つかったらソースを修正し“ｍａｋｅ”を実行して“＊．Ｈｅｘ”ファイルを作成し、ターゲットにダウンロードして、実行させ、再び確認する。

気の遠くなるような作業を何度も何度も繰り返し、仕様どうりのプログラムを完成させるわけです。

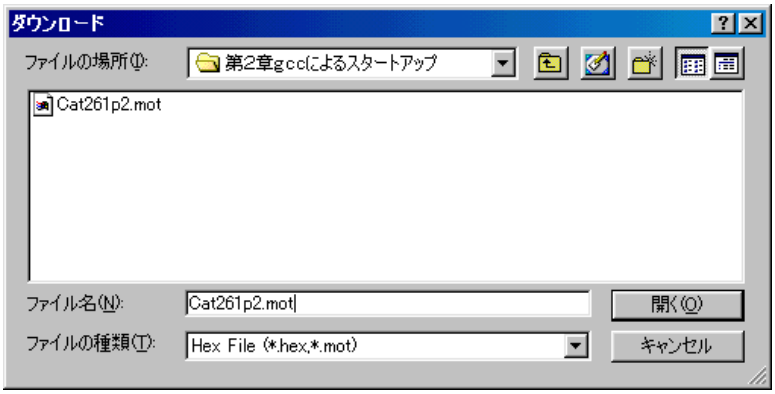
簡単ではありますが、プログラム開発の流れはだいたい掴んで頂けたかと思います。

第１章は、ここまでとし、次へのステップとして、Ｃ言語でのプログラム開発方法へと進めていきたいと思います。

第2章　ｇｃｃによるスタートアップ

　ここの章では、第１章の仕様そのままにして、ｍａｉｎ（）をＣ言語でプログラムした場合どのような仕組みでＨＥＸファイルが作られるのかを解説します。

まずは、ＤＥＦのダウンロードで



¥評価ボード¥第１部ポーリング編¥第２章ｇｃｃによるスタートアップ¥

にディレクトリの移動をしておいて下さい。

１．動かしてみましょう

　移動したディレクトリの中に、**“Ｃａｔ２６１ｐ２．ｍｏｔ”**というＨＥＸファイルがあります。これをダウンロードしてから、プログラム実行してみて下さい。

前章とまったく同じ動作のはずです。

それでは、プログラムリストを見てみましょう！

２．プログラムリスト

今後の移植性および役割分割のため、ソースを４ファイルに分割しました。

１）“ｖｅｃｔｏｒＡ．ａｓｍ”ベクターテーブルの定義です。前章のまま使用しました。

２）“Ｓｔａｒｔｕｐ．ｃ”スタートアップは、章が上がっても利用できるよう汎用化しました。

３）“Ｃａｔ２６１ｐ２．ｃ”メイン処理は、この章のメインコントロール部です。章が上がっていきますとこれをベースに追加していきます。

４）“ＰｏｌＰｉｏ２．ｃ”ＬＥＤ表示は、このサンプルソフトが動作しているかを目で確かめるために用意したソフトです。

ＬＥＤ点灯部分等は、後の章ＰＩＯ編で説明しますので、ここでの説明は省略します。

Ｃ言語の多モジュル化した場合に、定義文等を他のモジュールでも利用できるようにヘッダーファイルを２ファイル作成しました。

１）“ｉｏ２６１ｘ．ｈ”は、ＣＡＴ２６１（超小型マイコン）が使用しているＣＰＵ（Ｈ８Ｓ／２６１２）の内部Ｉ／Ｏをシンボル定義および、マクロ命令をまとめたファイルです。

２）“ＤｅｍｏＣｔｌ．ｈ”は、サンプルプログラムを見やすくするためのシンボル定義集とモジュール別に出来上がった関数のプロトタイプ宣言をまとめたファイルです。

章が上がることに、これをベースに追加していきます。

なお、［リストの説明］で前章と説明がダブル個所は省略します。

＊1）Ｃ言語によるスタートアップ（プログラム）

重要！！ このモジュールのリンク順番は、“ｖｅｃｔｏｒＡ．ｏ”（１番目）の次（２番目）に置いて下さい。

file “Startup.c”

```
/************************************************************************/
/* <サンプル>     ポーリング                                            */
/*                                                                      */
/*      <MOD>     Startup.c                                             */
/*      <タブ>    ４タブ編集      GNU/gcc                               */
/*                スタートアップ（CAT261-H8S/2612）                     */
/************************************************************************/
/************************************************************************/
/*      StartUp                                                         */
/************************************************************************/
void    StartUp()
{
    asm("mov.l  #0xffefbe,sp");         /* ｽﾀｯｸﾎﾟｲﾝﾀ 設定              */
    asm("jmp    __main      ");         /* void _main()                 */
}
/************************************************************************/
/*      non                                                             */
/************************************************************************/
#pragma interrupt
void    non()
{
}
```

［リストの説明］

内容説明については、前章の“ＳｔａｒｔｕｐＡ．ａｓｍ”と重複している部分もあります。

１～７行：

コメントです。

１１行：

電源ＯＮ時およびリセット時にプログラムはココから実行を始めるようにベクターテーブルに設定してあります。

１３行：

電源ＯＮ時およびリセット時にまず実行させる命令は、このスタックポインタの設定です。

定石ですので必ず守って下さい。

なお、理由はＣＰＵハードウェアマニュアルに記載されています。

１４行：

メインプログラムへのジャンプエントリです。

１９行：

割り込み例外処理を示す“＃ｐｒａｇｍａ　ｉｎｔｅｒｒｕｐｔ”定義になります。

この命令を置くことにより、関数の入口／出口に全レジスターの退避／復帰命令が挿入され、かつリターン命令が“ＲＴＥ”になります。

２０～２２行：

未使用割り込みのベクター値を置くためのプログラムです。

実際には、このプログラムが実行されることは通常状態である限りありえません。

ここまでがＣ言語記述での“スタートアップ”になるわけです。

＊2）メイン処理（プログラム）

file “Cat261p2.c”

```
/*****************************************************************************/
/*                                                                           */
/*  ＜サンプル＞  ポーリング                                                 */
/*                                                                           */
/*  ＜MOD＞       Cat261p2.c                                                 */
/*  ＜役割＞      main                                                       */
/*  ＜TAB＞       ４タブ編集                                                 */
/*  ＜保守ﾂｰﾙ＞  makefile 参照                                              */
/*  ＜使用ﾊｰﾄﾞ＞ CAT-261-H8S/2612 エーワン(株)                               */
/*                                                                           */
/*****************************************************************************/
#include    "io261x.h"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       Shift;              /* shift ﾊﾟﾀｰﾝ                      */
/*****************************************************************************/
/*      main()                                                               */
/*****************************************************************************/
void    _main(void)
{
    disable();                          /* PBC の割込の為                   */
    SYSCR = 0x21;                       /* ｼｽﾃﾑｺﾝﾄﾛｰﾗ 割込ﾓｰﾄﾞ 2           */
                                        /*             NMI 立下りエッジ     */
                                        /*             RAME 有効            */
    SoftWait1ms(400);                   /* 400msWait(ﾘｾｯﾄ遅延時間ﾊｰﾄﾞ)       */
                                        /* H-ﾃﾞﾊﾞｯｶﾞ<->Target 通信可能に成るまで*/
                                        /* の１回リトライ時間分待つ(20 回)   */
    MemInitial();                       /* ﾒﾓﾘ系初期化                      */
    IoInitial();                        /* I/O 系初期化                     */
    while(1) {
        SoftWait1ms(20);                /* 20msWait                         */
```

```
34:         RunRun();                       /* ｼﾌﾄLED点灯 OUTﾊﾞｯﾌｧｰにｾｯﾄ        */
35:         SigOutput();                    /* Signal Output Process(LED点灯)   */
36:     }
37: }
38: /*****************************************************************************/
39: /*      Ｍｅｍ初期化                                                         */
40: /*****************************************************************************/
41: void    MemInitial(void)
42: {
43:     Shift = 0;                          /* Led Disp Patan Initial           */
44:     PioMemInitial();                    /* PIO    Mem 初期化                */
45: }
46: /*****************************************************************************/
47: /*      Ｉ／Ｏ初期化                                                         */
48: /*****************************************************************************/
49: void    IoInitial(void)
50: {
51:     PioIoInitial();                     /* PIO    I/O 初期化                */
52: }
53: /*****************************************************************************/
54: /*      RunRun()        CPU 走行表示                                         */
55: /*****************************************************************************/
56: void    RunRun()
57: {
58:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
59:     PutOutPort(Shift,'=');
60: }
61: /*****************************************************************************/
62: /*      SoftWait1ms()   1ms単位 ｿﾌﾄﾀｲﾏｰ                                     */
63: /*****************************************************************************/
64: void    SoftWait1ms(Ushort ms)
65: {
66:     while(ms-- != 0) {
67:         Wait1ms();
68:     }
69: }
```

```
/************************************************************************/
/*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                */
/************************************************************************/
void    Wait1ms()
{
    asm("       push.w  r0                  ");
    asm("       mov.w   #5000,r0            ");     /* 5000*4=20000cyc  */
    asm(" wait:                             ");
    asm("       dec.w   #1,r0               ");     /* 1 clock          */
    asm("       bne     wait:16             ");     /* 3 clock          */
    asm("       pop.w   r0                  ");
}
```

［リストの説明］

１～１１行：

コメントです。

１２～１３行：

後で説明しますが、このテーマで使用する定義文をまとめたヘッダーファイルを使用することをコンパイラに教えるステートメントです。

１７行：

内部使用する変数の宣言文です。

“Uchar”は、“io261x.h”でキーワード宣言してあるため、“unsigned char”と同じ意味を持ちます。

長い文章を入力したく無い場合よく使います。(キー入力が苦手な日本人特有かも？)

変数名“Shift”の用途は、ＬＥＤ表示バッファーとして使用します。

２１行：

“_main()”関数の先頭を意味する宣言文です。

前に説明した“スタートアップ”から、ここにジャンプして来ます。

標準の“main()”では、“ｖｏｉｄ”タイプ宣言を許可していませんので、あえてアンダーバー“__”を付けて違う名前にしました。

２３行：

“ｉｏ２６１ｘ．ｈ”内に、“ｄｉｓａｂｌｅ ()”のマクロが組んであります。デバッガを使用するにあたり、ディセーブル状態は割り込み例外処理のプライオリティを“6”としています。

これにより、プライオリティ“7”以上の割り込みしか受け付けない命令になります。

つまり、Ｈ８Ｓ／２６１２のＰＣブレーク例外処理の割り込みプライオリティが“7”になっていますので、ＰＣブレーク例外処理を有効にさせるために、この命令必要です。

ＣＰＵタイプによりユーザープライオリティの設定要／不要があります。

１）Ｈ８／３６６４－――――――――――――――― 不要

２）Ｈ８Ｓ／２１３２・２１３４・２１３８――――― 不要

３）Ｈ８Ｓ／２２３８・２６１２・２６３３――――― 要（６以下）

４）ＳＨ／７０４５・７０５０・７０５１――――― 要（１４以下）

２４行：

Ｈ－ｄｅｂｕｇｇｅｒを使用する場合、システムコントローラに対する設定をこのように指示して下さい。(定石)

なお、ＣＰＵタイプにより設定内容が違います。

Ｈ８／３６６４の場合

１）ＮＭＩを立下りエッジにする。(デフォルト)

２）内部ＲＡＭ有効にする。(デフォルト)

Ｈ８Ｓ／２１３２・２１３４・２１３８の場合

１）ＮＭＩを立下りエッジにする。（デフォルト）

２）内部ＲＡＭ有効にする。（デフォルト）

Ｈ８Ｓ／２２３８・２２３８・２６３３の場合

１）割り込みモードを“2”にする。（モード“0”でも良い）

２）ＮＭＩを立下りエッジにする。（デフォルト）

３）内部ＲＡＭ有効にする。（デフォルト）

ＳＨ／７０４５・７０５０・７０５１の場合

１）ＮＭＩを立下りエッジにする。（デフォルト）

２）内部ＲＡＭ有効にする。（デフォルト）

２５行：

Ｈ－ｄｅｂｕｇｇｅｒを使用する場合、この４００ｍｓソフトタイマを入れておいて下さい。

理由は、リセット＋Ｍｏｎコマンド発行時にデバッガは、リセット出力後２００ｍｓ後に２０ｍｓ毎にターゲットにモニタが存在しているか調査しにいきます。（リトライ２０回）

この時にモニタが存在した場合、その時点でＰＣ値を０ｘ８００に戻しプログラム停止状態にします。

４００ｍｓタイマーがない場合は、この例ですとＩ／Ｏおよびメモリの初期化が進み、ユーザープログラムが最大２００ｍｓ走行後にプログラム停止状態になってからＰＣ値が０ｘ８００になります。

この動作が嫌な人は、ソフトタイマを付けてください。

ただし、ＤＥＦバージョン５．７０Ａよりリセット遅延防止タイマ未使用の指定ができますので、未使用に指定した場合は「２０ｕｓ以上」のソフトタイマ値になります。

３０行：

このテーマで使用する変数の初期化をまとめた関数です。

章が上がっていく度に追加されていきます。

電源ＯＮ時に使用ＲＡＭエリアをオールゼロにする関数をアセンブラで組む方法もありますが、今回は使用する変数一個一個を初期化する方法にしています。

３１行：

このテーマで使用するＩ／Ｏの初期化をまとめた関数です。

３２行：

電源ＯＦＦするまで無限ループをする先頭を意味するステートメントです。

“while(1) {---------}”までが無限ループ内になります。

３３行：

２０ｍｓ毎にＬＥＤをシフト表示させるための２０ｍｓのソフトタイマーです。

この関数を抜けて来るまで他の処理は一切しません。（もったいないことです）

３４行：

呼ばれる毎に変数“Shift”を１ビット左シフトし、ＬＥＤ表示のため、ｏｕｔバッファーにセットする関数を呼んでいます。

３５行：

ｏｕｔバッファーをポート出力する関数を呼んでいます。

３６行：

前に説明した“while(1)”の終わりを示す記号です。

３７行：

“main()”関数の終わりを示す記号です。

この２１行目から始まり、３６行目で終わる集まりを“関数”または“サブルーチン”と呼んでいます。

Ｃ言語で記述した場合は、この関数の集合体でプロジェクトを完成させています。

４１～４５行：

メモリ初期化の関数です。

４９～５２行：

Ｉ／Ｏ初期化の関数です。

５６～６０行：

動作表示モニタ用バッファを１ビット左シフトして、出力バッファにセットしています。

６４～６９行：

ｍｓｅｃ単位で指定するソフトタイマ関数です。

７３～８１行：

約１ｍｅｃのソフトタイマです。

ＣＰＵクロック＝２００００００Ｈｚですので、１０００Ｈｚ（１ｍｓ）で割りますとマシンサイクル＝２００００サイクルになります。

１ループ４サイクルですので、２００００÷４＝５０００ループになります。

前章ではアセンブラで作成してありましたが、参考にして頂くためインラインアセンブラでの記述してみました。

*3）ＰＩＯ関係（プログラム）

file “PolPio2.c” LED 表示（動作目視用）

```
/*****************************************************************************/
/*                                                                           */
/*  ＜サンプル＞  ポーリング                                                 */
/*                                                                           */
/*  ＜MOD＞       PolPio2.c                                                  */
/*  ＜役割＞      PIO 関係                                                   */
/*  ＜TAB＞       ４タブ編集                                                 */
/*  ＜保守ﾂｰﾙ＞   makefile 参照                                              */
/*  ＜使用ﾊｰﾄﾞ＞  CAT-261-H8S/2612 エーワン(株)                               */
/*                                                                           */
/*****************************************************************************/
#include    "io261x.h"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       OutPort;            /* Out  Port  Buffer                */
/*****************************************************************************/
/*      Ｍｅｍ初期化                                                         */
/*****************************************************************************/
void    PioMemInitial(void)
{
    OutPort = 0;                        /* OUT  Port  初期化                */
}
/*****************************************************************************/
/*      Ｉ／Ｏ初期化                                                         */
/*****************************************************************************/
void    PioIoInitial(void)
{
    P1DDR = 0xff;                       /* P1OB P0  全出力                  */
}
/*****************************************************************************/
/*      SigOutput   Signal Output Process(LED 出力)                          */
/*****************************************************************************/
```

```
34: /*****************************************************************************/
35: void    SigOutput()
36: {
37:     P1DR = ~OutPort;                    /* 0=点灯 1=消灯 のため              */
38: }
39: /*****************************************************************************/
40: /*      OutPortPut  出力バッファーにセット                                   */
41: /*****************************************************************************/
42: void    PutOutPort(Uchar patan,Uchar log)
43: {
44:     if (log == '=')      OutPort  = patan;
45:     else if (log == '|') OutPort |= patan;
46:     else if (log == '&') OutPort &= patan;
47: }
```

［リストの説明］

ＰＩＯについては後章で説明しますので、ここでは省略します。

＊1）ヘッダーファイル（1）

file “io261x.h”

```
  1: //***************************************************************************
  2: //
  3: //              CPU       =  H8S/2612
  4: //              MOD NAME  =  IO261x.H
  5: //              PROCESS   =  Ｉ／Ｏアドレス定義
  6: //              EDIT      =  ４タブ編集
  7: //
  8: //              Ver 1.00  2002-01-XX  M.Hasegawa
  9: //
 10: //***************************************************************************
 11: typedef unsigned char  Uchar;
 12: typedef unsigned short Ushort;
 13: typedef unsigned long  Ulong;
 14: //*********************************************************
 15: //*          インライン関数
 16: //*********************************************************
 17: #define disable()   asm("ldc.b  #0x6,exr")
 18: #define enable()    asm("ldc.b  #0x0,exr")
 19: //*********************************************************
 20: //          内部    I/O
 21: //*********************************************************
 22: #define MCR         (*(volatile char  *)0x0FFF800)          // HCAN
 23: #define GSR         (*(volatile char  *)0x0FFF801)
```

―――――――――― 途中省略 ――――――――――

```
536: #define PORTA       (*(volatile char  *)0x0FFFFB9)
537: #define PORTB       (*(volatile char  *)0x0FFFFBA)
538: #define PORTC       (*(volatile char  *)0x0FFFFBB)
539: #define PORTD       (*(volatile char  *)0x0FFFFBC)
540: #define PORTF       (*(volatile char  *)0x0FFFFBE)
```

［リストの説明］

１１～１３行：

“unsigned”長い予約語のため、このように予約語をキーワード宣言して使用しています。

使用したくないかたは、使用しなくてもかまいません。

ただし、サンプルソースは全部修正が必要になります。

１７行：

割り込み禁止のインラインアセム関数定義です。

Ｈ－ｄｅｂｕｇｇｅｒを使用した場合の割り込みディセーブルマスク値です。

モニタを外した場合でも、このままで問題無いはずですが、気になる方は、マスク値を“7”にして下さい。

１８行：

割り込み許可のインラインアセム関数定義です。

Ｈ－ｄｅｂｕｇｇｅｒを使用した場合の割り込みイネーブルマスク値です。

マスクレベルを変更されたいユーザーは、自由に変更して下さい。

２２～５４０行：

Ｈ８Ｓ／２６１２の内部Ｉ／Ｏレジスタをシンボル化しました。

ほとんどマニュアル記載どうりの名前にしましたが、重複部分は変更してありますので確認後、ご利用下さい。

*3）ヘッダーファイル（2）

file “DemoCtl.h”

```
/*****************************************************************************/
/*                                                                           */
/*  ＜役割＞      サンプルソフト特有の宣言                                   */
/*  ＜TAB＞       ４タブ編集                                                 */
/*                                                                           */
/*****************************************************************************/
/*****************************************************************************/
/*      マクロ                                                               */
/*****************************************************************************/
#define ON          0xaa                /* ﾌﾗｸﾞ      内部 ON  ﾌﾗｸﾞ          */
#define OFF         0                   /*           〃   OFF                */
/*****************************************************************************/
/*      プロトタイプ宣言                                                     */
/*****************************************************************************/
/*      Cat204p.c                   */
void    MemInitial(void);
void    IoInitial(void);
void    RunRun();
void    SoftWait1ms(Ushort  ms);
void    Wait1ms();
/*      PolPio.c                    */
void    PioMemInitial();
void    PioIoInitial();
void    SigOutput();
void    PutOutPort(Uchar patan,Uchar log);
```

［リストの説明］

１０～１１行：

内部で使用するフラグ数値をシンボル化しました。

直接数字を記述しますと、数字の意味が不明になるため、シンボル化しておくと便利です。

１５～２１行：

各モジュールで作成した関数をまとめてプロトタイプ宣言をしておきます。

プロトタイプ宣言をしておきますと、関数型および引数の型をコンパイラがチェックしてくれますので、是非宣言して下さい。

なお、これから章が上がるたびに関数が増えていきますので、当然プロトファイル宣言も比例して増えていき、このヘッダーファイルの内容も追加されていきますが、わざわざファイル名を変えて説明する必要も無いと思いますので、説明はこの章だけとさせてもらいます。

３．保守ツール（Ｍａｋｅｆｉｌｅ）

１）第１章のＭａｋｅｆｉｌｅとの違いを見る

```
7: TARGET   = Cat261p2.mot
8: OBJS     = vectorA.o   Startup.o Cat261p2.o PolPio2.o

28: Startup.o: Startup.c
29:   $(CC) $(CCFLAGS) -fomit-frame-pointer   $*.c
30:   $(AS) -o $*.o $*.s
```

［リストの説明］

７～８行：

プロジェクト名の変更およびモジュールの追加をしました。

２８～３０行：

今回は、スタートアップをＣ言語で記述しましたので、各関数ヘッドのフレームポインタ部分を削除するために、コンパイラスイッチ“－ｆｏｍｉｔ－ｆｒａｍｅ－ｐｏｉｎｔｅｒ”を追加しました。

理由としては、Ｓｔａｒｔｕｐ関数の最初の命令をスタックポインタの設定にするためにこのスイッチが必要になりました。

逆アセンブラーで確認して見てください。

後章からは、７，８行の変更と追加のみでアプリケーションに対応していきますので、“Ｍａｋｅｆｉｌｅ”についての説明については、この章で終了とします。

２）ｍａｋｅの実行

前章と同じように、２０ｍｓ毎にシフト表示しているのを、１００ｍｓ毎に変更してみましょう

“Ｃａｔ２６１ｐ２．ｃ”をエディタで開き

```
31:     while(1) {
32:         SoftWait1ms(20);                /* 20msWait                          */
33:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ        */
```

３２行目の“２０”を“１００”に変更してみてください。

```
32:         SoftWait1ms(100);               /* 100msWait                         */
```

変更後、保存終了して下さい。

保存終了が済みましたら、“ｍａｋｅ”を実行してみて下さい。

ｍａｋｅで新しく出来た“Ｃａｔ２６１ｐ２．ｍｏｔ”を評価ボードにダウンロードして、実行してみて下さい。

ＬＥＤのシフト表示スピードが遅くなりましたので、シフトしていく状況が目視で確認できるようになったはずです。

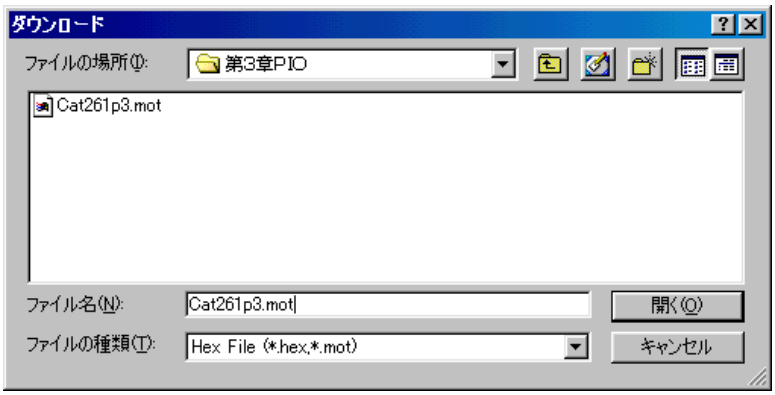
ここまで来ますと、Ｃ言語で開発する土台が出来あがりました。

次章は、評価ボードに付いている、トグルスイッチと押しボタンスイッチを使えるようにすることと、いままで使用していたＬＥＤ表示についての解説をします。

第3章　ＰＩＯ

　ここの章では、ＰＩＯイニシャルとＰＩＯ使用サンプルの解説を主におきたいと思います。
ＰＩＯの応用例として、入力（８点のトグルスイッチ、４点の押しボタンスイッチ）と出力（８点のＬＥＤ）と（ＬＣＤ表示ー第４章で説明します）です。

まずは、ＤＥＦのダウンロードで



¥評価ボード¥第１部ポーリング編¥第３章ＰＩＯ¥
にディレクトリの移動をしておいて下さい。

１．動かしてみましょう

後の説明進行のため、評価ボード下の**トグルスイッチ（８点）を全部を下（オフ）**にしておいて下さい。

移動したディレクトリの中に、**“Ｃａｔ２６１ｐ３．ｍｏｔ”**というＨＥＸファイルがあります。
これをダウンロードしてから、プログラム実行してみて下さい。

最初は、いままで通りの動作ですが、チョット仕様追加してあります。
評価ボードの右下の押しボタン（以後ＰＢと称します）**ＰＡ０**を押してみて下さい。
どうです？ＬＥＤ表示が消えたはずです。

ここで、評価ボード下のトグルスイッチ（以後ＳＷと称します）**Ｐ４０**を上（オン）にしてみて下さい。
どうです？ＬＥＤ［Ｐ１７］が点灯したはずです。

つまり、ＳＷをオンすると対角線上のＬＥＤを点灯させる仕様になっています。
とりあえず、**全ＳＷをオン／オフ**してみて下さい。

どうです？ＬＥＤが対角線上で点灯したはずです。

ここでもう１回、ＰＢ［ＰＡ０］を押してみて下さい。
最初に戻ってＬＥＤがパラパラ表示しているはずです。

これがすべての仕様ですが、下記の機能が追加されています。
１）ＰＩＯのイニシャル
２）ＳＷ，ＰＢの取りこみ（チャタリング取り付き）
３）ＳＷ入力処理
４）（ＬＥＤ表示処理）＜ーすでに使用していますが、未解説です。
５）モード制御
　機能の開始／停止をコントロールする部分が必要になってきましたので作成しましました。

動作はいたってシンプルですが、上記のプログラムを追加しなければ機能しないところが、マイコンプログラムの世話のかかるところです。

それでは、プログラムリストに沿って説明していきます。

２．プログラムリスト

このサンプルは、４ファイルの構成になっています。

“ｖｅｃｔｏｒＡ．ａｓｍ”　前章のまま使用したので解説を省略します。

“Ｓｔａｒｔｕｐ．ｃ”　前章のまま使用したので解説を省略します。

“ＰｏｌＰｉｏ．ｃ”　ＰＩＯ関係をまとめました。

“Ｃａｔ２６１ｐ３．ｃ”　メインコントロール部です。

１）ＰＩＯ関係

file　“PolPio.c”

```
/*****************************************************************************/
/*                                                                           */
/*  ＜サンプル＞  ポーリング                                                 */
/*                                                                           */
/*  ＜MOD＞       PolPio.c                                                   */
/*  ＜役割＞      PIO 関係                                                   */
/*  ＜TAB＞       ４タブ編集                                                 */
/*  ＜保守ﾂｰﾙ＞   makefile 参照                                              */
/*  ＜使用ﾊｰﾄﾞ＞  CAT-261-H8S/2612 エーワン(株)                               */
/*                                                                           */
/*****************************************************************************/
#include    <string.h>
#include    "io261x.h"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       InPort[2];          /* IN   Port  現 Buffer             */
        Uchar       UpPort[2];          /* IN   Port  立上り Buffer         */
        Uchar       InBack[2];          /* IN   Port  一ヶ前 Buffer         */
        Uchar       In20ms[2];          /* IN   Port  20ms 前 Buffe         */
        Uchar       InNows[2];          /* IN   Port  生 Buffer             */
        Uchar       OutPort;            /* Out  Port  Buffer                */
/*****************************************************************************/
/*      Ｍｅｍ初期化                                                        */
```

```
26: /*************************************************************************/
27: void    PioMemInitial(void)
28: {
29:     OutPort = 0;                        /* OUT  Port  初期化              */
30:     memset(InPort, 0,sizeof(InPort));   /* IN   Port  現Buffer            */
31:     memset(UpPort, 0,sizeof(UpPort));   /* IN   Port  立上りBuffer        */
32:     memset(InBack, 0,sizeof(InBack));   /* IN   Port  一ヶ前Buffer        */
33:     memset(In20ms, 0,sizeof(In20ms));   /* IN   Port  20ms前Buffe         */
34:     memset(InNows, 0,sizeof(InNows));   /* IN   Port  生Buffer            */
35: }
36: /*************************************************************************/
37: /*       Ｉ／Ｏ初期化                                                    */
38: /*************************************************************************/
39: void    PioIoInitial(void)
40: {
41:     P1DDR = 0xff;                       /* Port1   全出力[LED]            */
42:                                         /* Port4   入固定[SW]D0->D7       */
43:     PADDR = 0;                          /* PortA   入力  D0=PB[PA0]       */
44:     PCDDR = 0xc0;                       /* PortC   出力  D7=LCD-E         */
45:                                         /*         出力  D6=LCD-RS        */
46:                                         /*         入力  D5=PB[PC5]       */
47:                                         /*         入力  D2=PB[PC2]       */
48:     PFDDR = 0;                          /* PortF   入力  D7=PB[PF7]       */
49:     PDDDR = 0xff;                       /* PortD   全出力[LCD]            */
50: }
51: /*************************************************************************/
52: /*      SigInput  Signal Input Process(チャタ取り+状態検出+立上り検出)     */
53: /*************************************************************************/
54: void    SigInput()
55: {
56:     InNows[0] = ~PORT4;                 /* IN生 負論理 SW[P47->P40]       */
57:     if (PORTA & 0x1)  InNows[1] &= ~1;  /* IN生 負論理 PB[PF7,PC5,PC2,PA0] */
58:     else              InNows[1] |=  1;
59:     if (PORTC & 0x4)  InNows[1] &= ~2;
60:     else              InNows[1] |=  2;
61:     if (PORTC & 0x20) InNows[1] &= ~4;
```

```
62:     else              InNows[1] |=  4;
63:     if (PORTF & 0x80) InNows[1] &= ~8;
64:     else              InNows[1] |=  8;
65:     InPort[0] = In20ms[0] & InNows[0];  /* チャタ取り+状態検出[P47->P40]    */
66:     InPort[1] = In20ms[1] & InNows[1];  /*        〃           [PF7->PA0]    */
67:     UpPort[0] = (InPort[0] ^ InBack[0]) & InPort[0]; /* 立上検出[P47->P40] */
68:     UpPort[1] = (InPort[1] ^ InBack[1]) & InPort[1]; /*     〃   [PF7->PA0] */
69:     In20ms[0] = InNows[0];              /* 20ms 前 Buffe 記憶[P47->P40]      */
70:     In20ms[1] = InNows[1];              /*       〃        [PF7->PA0]        */
71:     InBack[0] = InPort[0];              /* １ヶ前記憶[P47->P40]              */
72:     InBack[1] = InPort[1];              /*       〃   [PF7->PA0]             */
73: }
74: /*************************************************************************/
75: /*      PioDemo()  ＰＩＯデモ                                            */
76: /*************************************************************************/
77: void    PioDemo()
78: {
79:     if (InPort[0] & 0x1)  OutPort = OutPort |   0x80;          /* SW[P40]  */
80:     else                  OutPort = OutPort & ~(0x80);
81:     if (InPort[0] & 0x2)  OutPort = OutPort |   0x40;          /* SW[P41]  */
82:     else                  OutPort = OutPort & ~(0x40);
83:     if (InPort[0] & 0x4)  OutPort = OutPort |   0x20;          /* SW[P42]  */
84:     else                  OutPort = OutPort & ~(0x20);
85:     if (InPort[0] & 0x8)  OutPort = OutPort |   0x10;          /* SW[P43]  */
86:     else                  OutPort = OutPort & ~(0x10);
87:     if (InPort[0] & 0x10) OutPort = OutPort |   0x8;           /* SW[P44]  */
88:     else                  OutPort = OutPort & ~(0x8);
89:     if (InPort[0] & 0x20) OutPort = OutPort |   0x4;           /* SW[P45]  */
90:     else                  OutPort = OutPort & ~(0x4);
91:     if (InPort[0] & 0x40) OutPort = OutPort |   0x2;           /* SW[P46]  */
92:     else                  OutPort = OutPort & ~(0x2);
93:     if (InPort[0] & 0x80) OutPort = OutPort |   0x1;           /* SW[P47]  */
94:     else                  OutPort = OutPort & ~(0x1);
95: }
96: /*************************************************************************/
97: /*      GetInPort() InPort[x]の読み取り                                  */
```

```
 98: /*****************************************************************************/
 99: Uchar   GetInPort(Uchar port)
100: {
101:     return(InPort[port]);
102: }
103: /*****************************************************************************/
104: /*      GetUpPort() UpPort[x]の読み取り                                      */
105: /*****************************************************************************/
106: Uchar   GetUpPort(Uchar port)
107: {
108:     return(UpPort[port]);
109: }
110: /*****************************************************************************/
111: /*      SigOutput   Signal Output Process(LED 出力)                          */
112: /*****************************************************************************/
113: void    SigOutput()
114: {
115:     P1DR = ~OutPort;                    /* 0=点灯 1=消灯 のため             */
116: }
117: /*****************************************************************************/
118: /*      OutPortPut  出力バッファーにセット                                   */
119: /*****************************************************************************/
120: void    PutOutPort(Uchar patan,Uchar log)
121: {
122:     if (log == '=')      OutPort  = patan;
123:     else if (log == '|') OutPort |= patan;
124:     else if (log == '&') OutPort &= patan;
125: }
```

［リストの説明］

１８～２３行：

このモジュールで使用する変数宣言です。

役割は、コメント参照して下さい。

２７～３５行：

このモジュールで使用する変数の初期化関数です。

メインのメモリ初期化の時に呼ばれます。

３９～４９行：

ＰＩＯの初期化関数です。

メインのＩ／Ｏ初期化の時に呼ばれます。

ポートｘデータディレクションレジスタに入出力方向をビットごとに指定します。

ビットを“１”にセットすると出力になります。

ビットを“０”にセットすると入力になります。

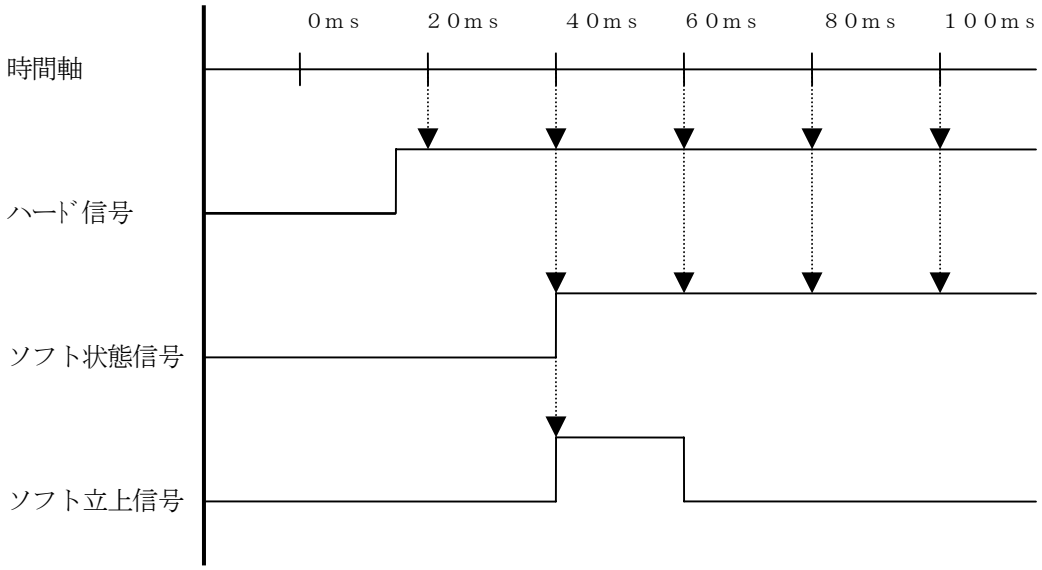
各ポートの割り振りは、コメントに記載されていますので、参考にしてください。

５４～７３行：

ＳＷおよびＰＢのチャタ取り＋状態検出＋立上り検出付きの入力関数です。

メインで２０ｍｓ毎に呼ばれます。

タイミングチャートで説明します。



ハード信号　　＝ＩｎＮｏｗｓ［ｎ］　ｎ＝０　ＳＷ［Ｐ４７－＞Ｐ４０］

ソフト状態信号　＝ＩｎＰｏｒｔ［ｎ］　ｎ＝１　ＰＢ［ＰＦ７－＞ＰＡ０］

ソフト立上信号　＝ＵｐＰｏｒｔ［ｎ］

２０ｍｓ前のハード信号　＝Ｉｎ２０ｍｓ［ｎ］

２０ｍｓ前のソフト状態信号＝ＩｎＢａｃｋ［ｎ］

の構成になっています。

ロジックを文章で説明するのは困難ですので、リストとタイミングチャートで解釈してみて下さい。

７７～９５行：

ＳＷ［Ｐ４０－＞Ｐ４７］のスイッチをオンすると、ＬＥＤ［Ｐ１０－＞Ｐ１７］を対角線上に点灯させるデモ関数です。

９９～１０２行：

ＳＷおよびＰＢのソフト状態信号を取得する関数です。

１０６～１０９行：

ＳＷおよびＰＢのソフト立ち上がり信号を取得する関数です。

１１３～１１６行：

出力バッファ“0utPort”をポート出力する関数です。

ハード的に、０＝点灯　１＝消灯のため、ここでＮＯＴにしています。

１２０～１２５行：

出力信号を出力バッファにセットする関数です。

2）メインコントロール

file “Cat261p3.c”

```
/*****************************************************************************/
/*                                                                           */
/*  ＜サンプル＞  ポーリング                                                 */
/*                                                                           */
/*  ＜MOD＞       Cat261p3.c                                                 */
/*  ＜役割＞      main                                                       */
/*  ＜TAB＞       ４タブ編集                                                 */
/*  ＜保守ﾂｰﾙ＞   makefile 参照                                              */
/*  ＜使用ﾊｰﾄﾞ＞ CAT-261-H8S/2612 エーワン(株)                                */
/*                                                                           */
/*****************************************************************************/
#include    "io261x.h"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       ModeStep;           /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
        Uchar       Shift;              /* shift ﾊﾟﾀｰﾝ                       */
/*****************************************************************************/
/*      _main()                                                              */
/*****************************************************************************/
void    _main(void)
{
    disable();                          /* PBC の割込の為                    */
    SYSCR = 0x21;                       /* ｼｽﾃﾑｺﾝﾄﾛｰﾗ 割込ﾓｰﾄﾞ 2            */
                                        /*             NMI 立下りエッジ      */
                                        /*             RAME 有効             */
    SoftWait1ms(400);                   /* 400msWait(ﾘｾｯﾄ遅延時間ﾊｰﾄﾞ)        */
                                        /* H-ﾃﾞﾊﾞｯｶﾞ<->Target 通信可能に成るまで*/
                                        /* の 1 回リトライ時間分待つ(20 回)   */
    MemInitial();                       /* ﾒﾓﾘ系初期化                       */
    IoInitial();                        /* I/O 系初期化                      */
    while(1) {
```

```
34:         SigInput();                 /* Signal Input  Process          */
35:         SoftWait1ms(20);            /*       ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り    */
36:         ModeCntrol();               /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                      */
37:         SigOutput();                /* Signal Output Process(LED 点灯)  */
38:     }
39: }
40: /*****************************************************************************/
41: /*      Ｍｅｍ初期化                                                         */
42: /*****************************************************************************/
43: void    MemInitial(void)
44: {
45:     ModeStep = 0;                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
46:     Shift = 0;                      /* Led Disp Patan Initial           */
47:
48:     PioMemInitial();                /* PIO    Mem 初期化                */
49: }
50: /*****************************************************************************/
51: /*      Ｉ／Ｏ初期化                                                         */
52: /*****************************************************************************/
53: void    IoInitial(void)
54: {
55:     PioIoInitial();                 /* PIO    I/O 初期化                */
56: }
57: /*****************************************************************************/
58: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                          */
59: /*****************************************************************************/
60: void    ModeCntrol()
61: {
62:     if (GetUpPort(1) & 0x1) {       /* PB[P30] ON?(立上)                */
63:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST */
64:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ   */
65:     }
66:     switch(ModeStep) {
67:     case 0:
68:         ModeStep++;
69:         break;
```

```
70:     case 1:
71:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ        */
72:         break;
73:     case 10:                            /*  PIO   TEST                      */
74:         ModeStep++;
75:         break;
76:     case 11:
77:         PioDemo();
78:         break;
79:     }
80: }
81:
82: /*************************************************************************/
83: /*      RunRun()        CPU 走行表示                                     */
84: /*************************************************************************/
85: void    RunRun()
86: {
87:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
88:     PutOutPort(Shift,'=');
89: }
90: /*************************************************************************/
91: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                 */
92: /*************************************************************************/
93: void    SoftWait1ms(Ushort ms)
94: {
95:     while(ms-- != 0) {
96:         Wait1ms();
97:     }
98: }
99: /*************************************************************************/
100: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait               */
101: /*************************************************************************/
102: void    Wait1ms()
103: {
104:     asm("       push.w  r0                  ");
105:     asm("       mov.w   #5000,r0            ");      /* 5000*4=20000cyc  */
```

```
106:    asm(" wait:                                  ");
107:    asm("       dec.w   #1,r0                    ");    /* 1 clock          */
108:    asm("       bne     wait:16                  ");    /* 3 clock          */
109:    asm("       pop.w   r0                       ");
110: }
```

［リストの説明］　　前章のメインコントロールから追加された部分だけ解説します。

１７行：

モード制御用に使用するコントロールステップ変数の宣言です。

３４行：

“ＰｏｌＰｉｏ．ｃ”で作成した、ＳＷとＰＢの入力関数を呼んでいます。

３６行：

モード制御する関数を呼んでいます。

４５行：

モード制御用に使用するコントロールステップ変数を初期化しています。

４８行：

“ＰｏｌＰｉｏ．ｃ”で使用する変数を初期化する関数を呼んでいます。

５５行：

ＰＩＯのＩ／Ｏ初期化する関数を呼んでいます。

６０～８０行：

モード制御の関数です。

ＰＢ［ＰＡ０］をオンしたら、ＰＩＯのデモ関数を呼ぶ仕組みになっています。

保守ツールについては、もうご理解したと思いますので、この章以降説明は省略します。

このサンプルで変更したい部分がありましたら各自変更をし、“ｍａｋｅ”を実行してみて下さい。

第4章　ＰＩＯの応用（ＬＣＤ）

　ここの章では、ＰＩＯの応用としてＬＣＤの表示関数を作成してみました。
表示機能を追加しますと、いろいろとしゃべることができますので表現が豊かになります。
この章は、応用例ですので各自リストを読み、何をやっているのかを理解していただくことが目的です。

まずは、ＤＥＦのダウンロードで

¥評価ボード¥第１部ポーリング編¥第４章ＰＩＯの応用（ＬＣＤ）¥
にディレクトリの移動をしておいて下さい。

１．動かしてみましょう

移動したディレクトリの中に、“Ｃａｔ２６１ｐ４．ｍｏｔ”というＨＥＸファイルがあります。
これをダウンロードしてから、プログラム実行してみて下さい。

どうです？　ＬＣＤにオープニングメッセージがでましたか？

仕様は、前章と同じですので、ＰＢ［ＰＡ０］を押してみて下さい。
表現が豊かになったと思います。

どのような仕組みでＬＣＤに表示させているか説明するためにプログラムリストに沿って説明をします。

２．プログラムリスト

このサンプルは、前章に１モジュール追加して、５ファイルの構成になっています。

“ｖｅｃｔｏｒＡ．ａｓｍ”	前章のまま使用したので解説を省略します。
“Ｓｔａｒｔｕｐ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＰｉｏ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＬｃｄ．ｃ”	追加したＬＣＤコントロールのモジュールです。
“Ｃａｔ２６１ｐ４．ｃ”	メインコントロール部です。

表示方法の仕組みを説明します。
ＬＣＤに表示したい場合、ＣＰＵ内部の表示バッファー“LcdBuf[2][16]”に表示データーを“LcdReq”に表示要求フラグをセットします。
そして、メインの１ループ毎で要求フラグを監視し、フラグが立っていた場合、メインより直接ＬＣＤに表示データを送る方式です。

１）ＬＣＤコントロール関係

file “PolLcd.c”

```
 1: /*****************************************************************************/
 2: /*                                                                           */
 3: /*  ＜サンプル＞  ポーリング                                                 */
 4: /*                                                                           */
 5: /*  ＜MOD＞       PolLcd.c                                                   */
 6: /*  ＜役割＞      LCD 関係                                                   */
 7: /*  ＜TAB＞       ４タブ編集                                                 */
 8: /*  ＜保守ﾂｰﾙ＞   Makefile 参照                                              */
 9: /*  ＜使用ﾊｰﾄﾞ＞  CAT-261-H8S/2612 エーワン(株)                               */
10: /*                                                                           */
11: /*****************************************************************************/
12: #include    <string.h>
13: #include    "io261x.h"
14: #include    "DemoCtl.h"
15: /*****************************************************************************/
16: /*       ＬＣＤ関係のマクロ                                                  */
17: /*****************************************************************************/
18: #define CLRCD       0x1                 /* LCD ｺﾏﾝﾄﾞ Disp Clear             */
```

```
#define EMDCD       0x6                 /*          ｴﾝﾄﾘｰﾓｰﾄﾞｾｯﾄ          */
#define FUKCD       0x38                /*          ﾌｧﾝｸｼｮﾝｾｯﾄ            */
#define DISON       0xc                 /*          表示ｵﾝ                */
#define DISOFF      0x8                 /*          表示ｵﾌ                */
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       LcdBuf[2][16];      /* LCD  表示  Buffer               */
        Uchar       LcdReq;             /*      表示  要求ﾌﾗｸﾞ             */
/*****************************************************************************/
/*      Ｍｅｍ初期化                                                         */
/*****************************************************************************/
void    LcdMemInitial(void)
{
    memset(LcdBuf[0],0x20,16);          /* LCD  表示  Buffer               */
    memset(LcdBuf[1],0x20,16);
    LcdReq = ON;                        /*      表示  要求ﾌﾗｸﾞ             */
}
/*****************************************************************************/
/*      LcdInitial()    LCDイニシャル                                        */
/*****************************************************************************/
void    LcdIoInitial()
{
    SoftWait1ms(45);                    /* Power On Wait 45ms              */
    LcdCmd(FUKCD);                      /* LCDｺﾏﾝﾄﾞ ﾌｧﾝｸｼｮﾝｾｯﾄ(1)          */
    SoftWait1ms(5);                     /*          5ms Wait               */
    LcdCmd(FUKCD);                      /* LCDｺﾏﾝﾄﾞ ﾌｧﾝｸｼｮﾝｾｯﾄ(2)          */
    SoftWait10us(10);                   /*          100us Wait             */
    LcdCmd(FUKCD);                      /* LCDｺﾏﾝﾄﾞ ﾌｧﾝｸｼｮﾝｾｯﾄ(3)          */
    LcdCmd(FUKCD);                      /* LCDｺﾏﾝﾄﾞ ﾌｧﾝｸｼｮﾝｾｯﾄ(4)          */
    LcdDispOn();                        /* LCD      表示ｵﾝ                */
    LcdDispClear();                     /* LCD      表示ｸﾘｱ               */
    LcdCmd(EMDCD);                      /* LCDｺﾏﾝﾄﾞ ｴﾝﾄﾘｰﾓｰﾄﾞｾｯﾄ          */
    SoftWait10us(4);                    /*          40us Wait              */
}
/*****************************************************************************/
```

```
/*      AllLcdDisp()        全画面表示                                  */
/*******************************************************************/
void    AllLcdDisp()
{
    if (LcdReq == ON) {             /* 表示要求                        */
        LcdReq = OFF;               /*   〃    OFF にするのはココだけ   */
        LcdDispOff();
        GotoxyDisp(0,0,LcdBuf[0]);  /* 1 行目                          */
        GotoxyDisp(0,1,LcdBuf[1]);  /* 2 行目                          */
        LcdDispOn();
    }
}
/*******************************************************************/
/*      GotoxyMemSet()      画面表示バッファーにセット                  */
/*******************************************************************/
void    GotoxyMemSet(Uchar x,Uchar y,Uchar *str)
{
    Uchar   *ptr;

    ptr = &LcdBuf[y][x];
    while(*str != 0) {*ptr++ = *str++;}
    LcdReq = ON;                    /* 表示要求 ON にするのはココだけ   */
}
/*******************************************************************/
/*      GotoxyDisp()        カーソル移動＋表示                          */
/*******************************************************************/
void    GotoxyDisp(Uchar x,Uchar y,Uchar *str)
{
    Gotoxy(x,y);                    /* カーソル移動                    */
    while(*str != 0) {
        LcdPutch(*str++);           /* １文字表示                      */
    }
}
/*******************************************************************/
/*      Gotoxy()            カーソル移動                                */
/*******************************************************************/
```

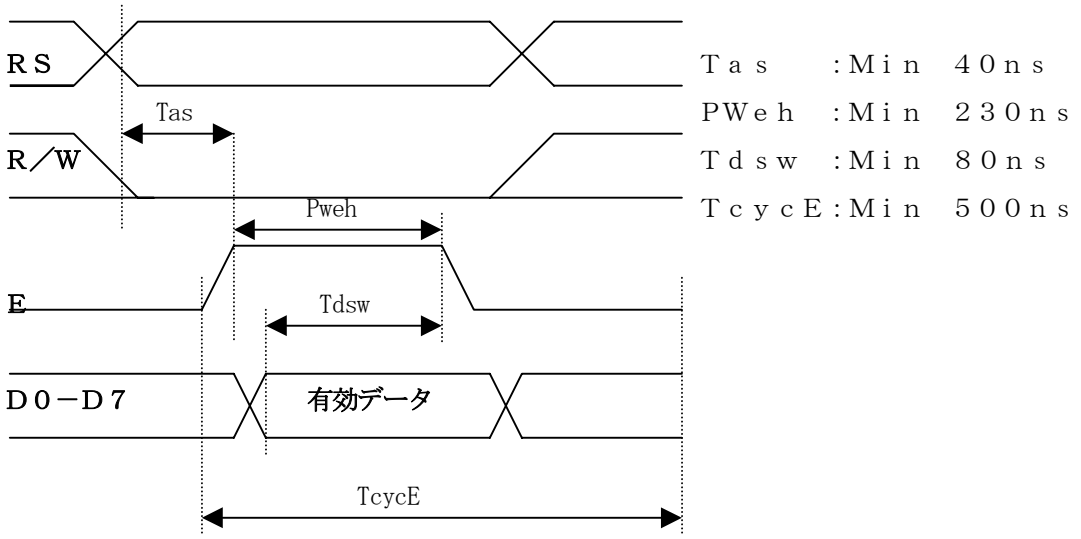
```
 91: void    Gotoxy(Uchar x,Uchar y)
 92: {
 93:     Uchar   ramadr;
 94:
 95:     if (y == 0) ramadr = 0;             /* DDRAM ｱﾄﾞﾚｽ計算                  */
 96:     else        ramadr = 0x40;
 97:     ramadr += x;
 98:     LcdCmd(ramadr | 0x80);              /* LCD ｺﾏﾝﾄﾞ DDRAM ｱﾄﾞﾚｽｾｯﾄ          */
 99:     SoftWait10us(4);                    /*          40us Wait               */
100: }
101: /*************************************************************************/
102: /*      LcdDispOn/Off()     表示オン/オフ コントロール                       */
103: /*************************************************************************/
104: void    LcdDispOn()
105: {
106:     LcdCmd(DISON);                      /* LCD ｺﾏﾝﾄﾞ 表示ｵﾝ                */
107:     SoftWait10us(4);                    /*          40us Wait               */
108: }
109: void    LcdDispOff()
110: {
111:     LcdCmd(DISOFF);                     /* LCD ｺﾏﾝﾄﾞ 表示ｵﾌ                */
112:     SoftWait10us(4);                    /*          40us Wait               */
113: }
114: /*************************************************************************/
115: /*      LcdDispClear()      表示クリア コントロール                          */
116: /*************************************************************************/
117: void    LcdDispClear()
118: {
119:     LcdCmd(CLRCD);                      /* LCD ｺﾏﾝﾄﾞ Disp Clear           */
120:     SoftWait10us(164);                  /*          1.64ms Wait             */
121: }
122: /*************************************************************************/
123: /*      LcdPutch()  LCD DDRAM Char Data Write                              */
124: /*************************************************************************/
125: void    LcdPutch(Uchar data)
126: {
```

```
127:     PCDR |= 0x40;                          /* LCD  RS     ON                  */
128:     if (data < 0x20) data = 0x20;
129:     LcdCmd(data);
130:     PCDR &= ~0x40;                         /* LCD  RS     OFF                 */
131:     SoftWait10us(4);                       /*          40us Wait              */
132: }
133: /*****************************************************************************/
134: /*      LcdCmd()    LCD command OUT                                          */
135: /*****************************************************************************/
136: void    LcdCmd(Uchar cmd)
137: {
138:     PDDR  = cmd;                           /* LCD  Data                      */
139:     PCDR |= 0x80;                          /*      E      ON                 */
140:     PCDR &= ~0x80;                         /*      E      OFF                */
141:
142: }
```

リストの説明に入る前に、ＬＣＤコントローラ関係資料を添付します。

信号名		機能
Ｄ０－Ｄ７	入出力	８本のデータバス。トライステート双方向性 この線を通してデータ・コマンドのやり取りをします。
Ｅ	入力	動作起動信号。データの書き込みおよび読み出しの起動をかけます。
Ｒ／Ｗ	入力	読み出し（Ｒ）／書き込み（Ｗ）の選択信号 “１”：読み出し　“０”：書きこみ
ＲＳ	入力	レジスタを選択する信号。 “０”：インストラクションレジスタ（Ｗ） ビジィフラグ、アドレスカウンタ（Ｒ） “１”：データレジスタ
ＶＯ	電源	液晶表示駆動用電源、ＶＯを変えることにより画面の濃淡を変化させることができます
ＶＤ	電源	＋５Ｖ
ＶＳ	電源	グランド端子：０Ｖ

図［４－２－１］　端子機能



図［４－２－２］　書込みタイミング

ＤＤ　ＲＡＭアドレスと表示桁の対応関係

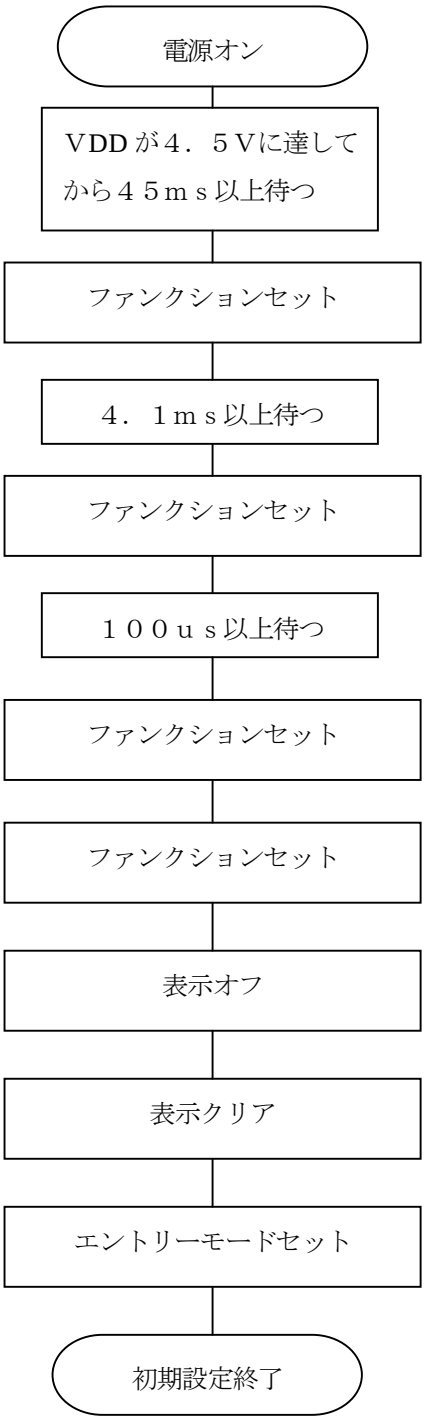
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
１行目	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
２行目	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

図［４－２－３］　ＤＤ　ＲＡＭアドレスマップ

インストラクション	コード			機能	実行時間 (max)
	RS	R/W	D7 D6 D5 D4 D3 D2 D1 D0		
表示クリア	0	0	0 0 0 0 0 0 0 1	全表示クリア後、カーソルをホーム位置へ戻します。	1.64ms
カーソルホーム	0	0	0 0 0 0 0 0 1 ＊	カーソルをホーム位置へ戻します、シフトしていた表示も戻ります。DDRAM の内容は変化しません	1.64ms
エントリーモードセット	0	0	0 0 0 0 0 1 I/D S	データの書き込みおよび読み出し時に、カーソルの進む方向、表示をシフトかの設定をする。	40us
表示オン／オフコントロール	0	0	0 0 0 0 1 D C B	全表示オン／オフ（D） ｶｰｿﾙのｵﾝ/ｵﾌ（C） ｶｰｿﾙ位置のﾌﾞﾘﾝｸ（B）	40us
カーソル／表示シフト	0	0	0 0 0 1 S/C R/L ＊＊	DD RAM の内容を変えずに、カーソルの移動と、表示シフトをします。	40us
ファンクションセット	0	0	0 0 1 DL N F ＊ ＊	ｲﾝﾀｰﾌｪｰｽﾃﾞｰﾀ長（ＤＬ） ﾃﾞｭｰﾃｨ（Ｎ）、文字ﾌｫﾝﾄ（Ｆ）を設定します。	40us
ＤＤ　ＲＡＭアドレスセット	0	0	1―――ADD―――	ＤＤ　ＲＡＭのアドレスをセットします。 以後のデータはＤＤＲＡＭのデータになります。	40us

Ｉ／Ｄ＝１：インクリメント　＝０：デクリメント
Ｓ　＝１：表示シフトする　＝０：しない
Ｄ　＝１：表示オン　＝０：表示オフ
Ｃ　＝１：カーソルオン　＝０：カーソルオフ
Ｂ　＝１：ブリンクオン　＝０：ブリンクオフ
Ｓ／Ｃ＝１：表示シフト　＝０：カーソル移動
Ｒ／Ｌ＝１：右シフト　＝０：左シフト
ＤＬ　＝１：８ビット　＝０：４ビット
Ｎ　＝１：１／１６デューティ　＝０：１／８、１／１１
Ｆ　＝１：５ｘ１０ドット　＝０：５ｘ７ドット
＊　＝無効ﾋﾞｯﾄ
ＡＤＤ＝ＤＤＲＡＭアドレス

図［４－２－４］　インストラクション一覧



図［４－２－５］　ＬＣＤ初期設定手順

［リストの説明］

１８～２２行：

ＬＣＤインストラクションコードのシンボル定義です。

２６行：

ＣＰＵ内部のＬＣＤバッファーの宣言です。

２７行：

ＬＣＤ表示要求フラグの宣言です。

３１～３６行：

このモジュールで使用する変数の初期化関数です。

メインのメモリ初期化の時に呼ばれます。

４０～５３行：

ＬＣＤ初期設定する関数です。

メインのＩ／Ｏ初期化の時に呼ばれます。

図［４－２－５］を参照して下さい。

５７～６６行：

ＬＣＤ表示要求フラグが立っていた場合、直接ＬＣＤに表示データを全画面転送する関数です。

常にメインループ１回に１回呼ばれます。

７０～７７行：

ＬＣＤ表示バッファーにセットする関数です。

ここで表示要求フラグを立てています。

８１～８７行：

ＬＣＤに直接、文字列転送する関数です。

９１～１００行：

ＬＣＤのカーソルを移動させる関数です。

１０４～１１３行：

ＬＣＤの表示をオン／オフさせる関数です。

１１７～１２１行：

ＬＣＤ全画面をクリアする関数です。

１２５～１３２行：

ＬＣＤのＤＤ　ＲＡＭに１バイト転送する関数です。

１３６～１４２行：

ＬＣＤのインストラクションコードを発行する関数です。

2）メインコントロール

file “Cat261p4.c”

```
/*****************************************************************************/
/*                                                                           */
/*  ＜サンプル＞  ポーリング                                                 */
/*                                                                           */
/*  ＜MOD＞       Cat261p4.c                                                 */
/*  ＜役割＞      main                                                       */
/*  ＜TAB＞       ４タブ編集                                                 */
/*  ＜保守ﾂｰﾙ＞  Makefile 参照                                              */
/*  ＜使用ﾊｰﾄﾞ＞ CAT-261-H8S/2612 エーワン(株)                               */
/*                                                                           */
/*****************************************************************************/
#include    "io261x.h"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       ModeStep;           /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
        Uchar       Shift;              /* shift ﾊﾟﾀｰﾝ                       */
/*****************************************************************************/
/*      _main()                                                              */
/*****************************************************************************/
void    _main(void)
{
    disable();                          /* PBC の割込の為                    */
    SYSCR = 0x21;                       /* ｼｽﾃﾑｺﾝﾄﾛｰﾗ 割込ﾓｰﾄﾞ 2            */
                                        /*             NMI 立下りエッジ      */
                                        /*             RAME 有効             */
    SoftWait1ms(400);                   /* 400msWait(ﾘｾｯﾄ遅延時間ﾊｰﾄﾞ)        */
                                        /* H-ﾃﾞﾊﾞｯｶﾞ<->Target 通信可能に成るまで*/
                                        /* の 1 回リトライ時間分待つ(20 回)    */
    MemInitial();                       /* ﾒﾓﾘ系初期化                       */
    IoInitial();                        /* I/O 系初期化                      */
    while(1) {
```

```
34:         SigInput();                     /* Signal Input  Process           */
35:         SoftWait1ms(20);                /*        ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り     */
36:         ModeCntrol();                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                       */
37:         AllLcdDisp();                   /* LCD    全画面表示               */
38:         SigOutput();                    /* Signal Output Process(LED 点灯)  */
39:     }
40: }
41: /*****************************************************************************/
42: /*      Ｍｅｍ初期化                                                         */
43: /*****************************************************************************/
44: void    MemInitial(void)
45: {
46:     ModeStep = 0;                       /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                 */
47:     Shift = 0;                          /* Led Disp Patan Initial          */
48:
49:     PioMemInitial();                    /* PIO    Mem 初期化               */
50:     LcdMemInitial();                    /* LCD    Mem 初期化               */
51: }
52: /*****************************************************************************/
53: /*      Ｉ／Ｏ初期化                                                         */
54: /*****************************************************************************/
55: void    IoInitial(void)
56: {
57:     PioIoInitial();                     /* PIO    I/O 初期化               */
58:     LcdIoInitial();                     /* LCD    I/O 初期化               */
59: }
60: /*****************************************************************************/
61: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                           */
62: /*****************************************************************************/
63: void    ModeCntrol()
64: {
65:     if (GetUpPort(1) & 0x1) {           /* PB[PA0] ON?(立上)                */
66:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST    */
67:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ      */
68:     }
69:     switch(ModeStep) {
```

```
70:     case 0:
71:         GotoxyMemSet(0,0,"CAT261&AH6000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ       */
72:         GotoxyMemSet(0,1,"Polling    [PA0]");
73:         ModeStep++;
74:         break;
75:     case 1:
76:         RunRun();                       /* ｼﾌﾄLED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ        */
77:         break;
78:     case 10:                            /*  PIO   TEST                      */
79:         GotoxyMemSet(0,0,"PIO             ");
80:         GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
81:         ModeStep++;
82:         break;
83:     case 11:
84:         PioDemo();
85:         break;
86:     }
87: }
88: /*************************************************************************/
89: /*      RunRun()        CPU 走行表示                                     */
90: /*************************************************************************/
91: void    RunRun()
92: {
93:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
94:     PutOutPort(Shift,'=');
95: }
96: /*************************************************************************/
97: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                 */
98: /*************************************************************************/
99: void    SoftWait1ms(Ushort ms)
100: {
101:     while(ms-- != 0) {
102:         Wait1ms();
103:     }
104: }
105: /*************************************************************************/
```

```
106: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                  */
107: /*****************************************************************************/
108: void    SoftWait10us(Ushort us)
109: {
110:     while(us-- != 0) {
111:         Wait10us();
112:     }
113: }
114: /*****************************************************************************/
115: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                   */
116: /*****************************************************************************/
117: void    Wait1ms()
118: {
119:     asm("       push.w  r0                      ");
120:     asm("       mov.w   #5000,r0                ");      /* 5000*4=20000cyc  */
121:     asm(" wait:                                 ");
122:     asm("       dec.w   #1,r0                   ");      /* 1 clock          */
123:     asm("       bne     wait:16                 ");      /* 3 clock          */
124:     asm("       pop.w   r0                      ");
125: }
126: /*****************************************************************************/
127: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                  */
128: /*****************************************************************************/
129: void    Wait10us()
130: {
131:     asm("       push.w  r0                      ");
132:     asm("       mov.w   #50,r0                  ");      /* 50*4=200cyc      */
133:     asm(" wait1:                                ");
134:     asm("       dec.w   #1,r0                   ");      /* 1 clock          */
135:     asm("       bne     wait1:16                ");      /* 3 clock          */
136:     asm("       pop.w   r0                      ");
137: }
```

［リストの説明］　　前章のメインコントロールから追加された部分だけ解説します。

３７行：

ＬＣＤ全画面表示関数を呼んでいます。

５０行：

“ＰｏｌＬｃｄ．ｃ”で使用する変数を初期化する関数を呼んでいます。

５８行：

ＬＣＤの初期設定をする関数を呼んでいます。

７１～７２行：

ＬＣＤにオープニングメッセージを表示させるための関数を呼んでいます。

７９～８０行：

ＬＣＤにＰＩＯモードメッセージを表示させるための関数を呼んでいます。

これで、この章のリスト説明は終わりです。

ご理解いただけたでしょうか？　プログラム記述は個性がヒジョウにでるもので、なれない記述だと読みにくいと思います。

私自身も、他人の書いたプログラムを読むには、かなりのエネルギーが必要です。

しかし、プログラム記述の標準仕様ができない限り、この問題はプログラマに付きまといます。

根気に読み続けて頂き理解してもらいたいと思います。

次は、タイマ／カウンタ使用例の解説へと進みます。

第5章　タイマ／カウンタ

　ここの章では、タイマ／カウンタのイニシャルとタイマ使用サンプルの解説を主におき、
応用例として評価ボードに外付けしたブザーを利用したいと思います。
ＣＮ１３－８Ｂ（ＰＢ０／ＴＩＯＣＡ３）にブザーを接続します。
タイマーのＰＷＭモードを利用し、指定周波数のパルス出力をしますとブザーを色々な音階で鳴らすことができますので、この仕組みを使ったサンプルを作成していきたいと思います。

まずは、ＤＥＦのダウンロードで
¥評価ボード¥第１部ポーリング編¥第５章タイマ／カウンタ¥

にディレクトリの移動をしておいて下さい。

１．動かしてみましょう

移動したディレクトリの中に、“Ｃａｔ２６１ｐ５．ｍｏｔ”というＨＥＸファイルがあります。これをダウンロードしてから、プログラム実行してみて下さい。

どうです？　ＬＣＤにオープニングメッセージがでましたか？

仕様は、前章に追加したかたちになります。ＰＢ［ＰＡ０］を押してみて下さい。

どうです？　ＬＣＤ表示の左上に“ＰＩＯ”と表示したはずです。

ここでもう１回、ＰＢ［ＰＡ０］を押してみて下さい。

どうです？　ＬＣＤ表示の左上に“Ｔｉｍｅｒ／Ｃｏｕｎｔｅｒ”と表示したはずです。

ここが、この章の追加サンプルプログラム部分です。

ここで、ＰＢ［ＰＦ７］を押してみて下さい。

どうです？　ブザーが鳴り、ＬＣＤ表示の左下に“２００Ｈｚ”と表示したはずです。

ここで、数回ＰＢ［ＰＦ７］を押してみて下さい。

どうです？　ブザーの音が高くなり、ＬＣＤに周波数を表示しているはずです。

ここで、ＰＢ［ＰＣ５］を押してみて下さい。周波数が下がったはずです。

ここでの操作仕様は、
ＰＢ［ＰＡ０］　モード開始／終了
ＰＢ［ＰＦ７］　周波数を１００Ｈｚ上げます（ＭＡＸ　１１００Ｈｚ）
ＰＢ［ＰＣ５］　周波数を１００Ｈｚ下げます（ＭＩＮ　２００Ｈｚ）
です。
ブザーを鳴らすだけでなく、ＣＮ１３－８Ｂの信号をシンクロで見るのも面白いかもしれません。
それでは、プログラムリストを見てみましょう！

２．プログラムリスト

このサンプルは、前章に２モジュール追加して、７ファイルの構成になっています。

“ｖｅｃｔｏｒＡ．ａｓｍ”	前章のまま使用したので解説を省略します。
“Ｓｔａｒｔｕｐ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＰｉｏ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＬｃｄ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＴｉｍｅ．ｃ”	Ｔｉｍｅｒコントロールのモジュールです。
“ＣａｔＳｕｂ．ｃ”	共通サブルーチンのモジュールです。
“Ｃａｔ２６１ｐ５．ｃ”	メインコントロール部です。

１）Ｔｉｍｅｒコントロール関係

file “PolTime.c”

```
 1: /*****************************************************************************/
 2: /*                                                                           */
 3: /*  ＜サンプル＞  ポーリング                                                 */
 4: /*                                                                           */
 5: /*  ＜MOD＞       PolTime.c                                                  */
 6: /*  ＜役割＞      タイマ関係                                                 */
 7: /*  ＜TAB＞       ４タブ編集                                                 */
 8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                              */
 9: /*  ＜使用ﾊｰﾄﾞ＞  CAT-261-H8S/2612 エーワン(株)                               */
10: /*                                                                           */
11: /*****************************************************************************/
12: #include    <string.h>
13: #include    "io261x.h"
14: #include    "DemoCtl.h"
15: /*****************************************************************************/
16: /*      ブザー（タイマ）関係のマクロ                                         */
17: /*****************************************************************************/
18: #define BZMIN       200                 /* Buzzer   Min 200Hz              */
19: #define BZMAX       1100                /*          Max 1100Hz             */
20: /*****************************************************************************/
21: /*      変数宣言                                                             */
22: /*****************************************************************************/
```

```
23:         short       BuzzerHz;           /* Buzzer Hz                        */
24: /*****************************************************************************/
25: /*      Ｍｅｍ初期化                                                         */
26: /*****************************************************************************/
27: void    TimMemInitial(void)
28: {
29:     BuzzerHz = 0;                       /* Buzzer Hz                        */
30: }
31: /*****************************************************************************/
32: /*      Ｉ／Ｏ初期化                                                         */
33: /*****************************************************************************/
34: void    TimIoInitial(void)
35: {
36:     MSTPCRA &= ~0x20;                   /* ﾓｼﾞｭｰﾙｽﾄｯﾌﾟ TPU ｽﾀｰﾄ             */
37:     TSTR = 0;                           /* Buzzer TPU_3 ここでは停止        */
38: }
39: /*****************************************************************************/
40: /*      Buzzer   TPU_3  TIOCA3 出力に Buzzer 接続  sys = 20,000,000Hz        */
41: /*****************************************************************************/
42: void    Buzzer(Ushort hz)
43: {
44:     Ushort   cyc;
45:
46:     if ((hz >= BZMIN) && (hz <= BZMAX)) {   /* Buzzer ON                    */
47:         TCR_3 = 0x21;                   /* CNTL TGRA ｺﾝﾍﾟｱﾏｯﾁで TCNT ｸﾘｱ   001 */
48:                                         /*      立上りｴｯｼﾞ                 00  */
49:                                         /*      sys/4(5,000,000Hz)         001 */
50:         TMDR_3 = 0xc2;                  /* MODE Default                    1100 */
51:                                         /*      PWM ﾓｰﾄﾞ 1                  0010 */
52:         TIORH_3 = 3;                    /* I/O  TGRB 未使用                0000 */
53:                                         /*      TIOCA3 ﾏｯﾁﾄｸﾞﾙ出力         0011 */
54:         cyc = 5000000 / hz;             /* 周期の計算                       */
55:         TGRA_3 = cyc / 2;               /* 周期設定/2                       */
56:         TCNT_3 = 0;                     /* ｶｳﾝﾄｸﾘｱ                          */
57:         TSTR  |= 0x8;                   /* TCNT_3 ｽﾀｰﾄ                      */
58:     }
```

```
59:     else {                          /* Buzzer OFF                   */
60:         TCNT_3 = 0;                 /* ｶｳﾝﾄｸﾘｱ                      */
61:         TSTR  &= ~0x8;              /* TCNT_3 ｽﾄｯﾌﾟ                 */
62:   }
63: }
64: /*****************************************************************************/
65: /*      TimerDemo   Timer デモ                                               */
66: /*****************************************************************************/
67: void    TimerDemo()
68: {
69:     Uchar   port;
70:     Uchar   dec[4+1];
71:
72:     port = GetUpPort(1);            /* PB[PA0]->PB[PF7]             */
73:     if (port & 0xc) {               /* PB[PC5] | PB[PF7] ON ?       */
74:         if (port & 0x4) {           /* PB[-PC5] ON-立上り            */
75:             BuzzerHz -= 100;
76:         }
77:         else if (port & 0x8) {      /* PB[+PF7] ON-立上り            */
78:             BuzzerHz += 100;
79:         }
80:         if (BuzzerHz < BZMIN) BuzzerHz = BZMIN;
81:         if (BuzzerHz > BZMAX) BuzzerHz = BZMAX;
82:         Buzzer(BuzzerHz);
83:         Bin2AdecN(dec,BuzzerHz,4);  /* 表示用データ作成             */
84:         GotoxyMemSet(0,1,dec);
85:     }
86: }
```

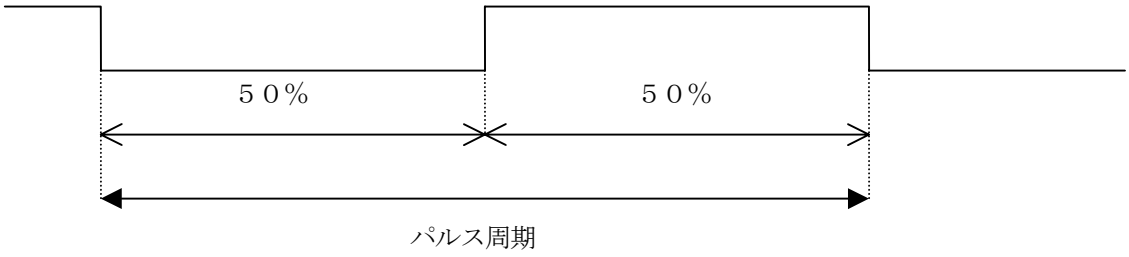
リストの説明に入る前に、PWMモードの説明をします。
このサンプルでは、ＴＰＵ＿３を使用しましたので、ＴＰＵ＿３について説明します。

（1）使用内部レジスタ

シンボル	機能	Ｃソースライン
ＭＳＴＰＣＲＡ	ＴＰＵのモジュールストップモードを解除する	３６行
ＴＳＴＲ	タイマカウンタの動作／停止を設定する	３７・５７・６１行
ＴＣＲ＿３	ＴＣＮＴの入力クロックおよびクリア要因を設定する	４７行
ＴＭＤＲ＿３	ＰＷＭモードの設定	５０行
ＴＩＯＲ＿３	コンペアマッチＡ発生時の出力パルスレベルの設定	５２行
ＴＧＲＡ＿３	出力パルスの１／２周期を設定する	５５行
ＴＣＮＴ＿３	カウンターとして使用する	５６行

表［５－２－１］　ＴＰＵ＿３内部レジスタ

（2）パルス出力仕様



・ＰＷＭ１モード使用
・コンペアマッチ時、カウンタクリア機能
・トグル出力機能
・ＴＣＮＴ＿３のクロックをｓｙｓ／４でカウント（ｓｙｓ＝２０ＭＨｚ）

（3）周期計算

システムクロック＝２０ＭＨｚとして、ＴＣＮＴの内部カウントクロックをｓｙｓ／４としましたので、

２０ＭＨｚ　÷　４　＝　５．０ＭＨｚ（内部クロック）

ＴＣＮＴは、ｗｏｒｄサイズ（１６ビット）ですので、最大６５５３５まで設定できるとして、

５．０ＭＨｚ　÷　（６５５３５　×　２）　＝　約３９Ｈｚ　（最小周波数）
５．０ＭＨｚ　÷　（１　　×　２）　＝　２．５ＭＨｚ（最大周波数）

になります

［リストの説明］

１８行：

ブザー出力周波数の最小値の定義です。

１９行：

ブザー出力周波数の最大値の定義です。

２３行：

出力周波数を記憶しておく変数の宣言です。

２７～３０行：

このモジュールで使用する変数の初期化関数です。

メインのメモリ初期化の時に呼ばれます。

３４～３８行：

タイマ／カウンタを初期化する関数です。

メインのＩ／Ｏ初期化の時に呼ばれます。

重要　モジュールストップコントロールレジスタのＴＰＵをオフ（開始）することをお忘れなく！

ただし、ここではＴＰＵ＿３未動作状態で初期化しておきます。

４２～６３行：

指定周波数のパルスをブザーに出力する関数です。ＴＰＵ＿３使用します。

指定周波数が、範囲外だった場合、ブザーを止める仕組みになっています。

指定周波数が、範囲内だった場合は、指定周波数になるように周期計算をし、デューティ５０％になるように、PWM設定のトグル出力を使用します。

１）ＴＣＮＴの内部カウントクロックプリスケール設定およびクリア要因の設定

２）タイマモードの設定（PWMモード１）

３）ＴＩＯＲのコンペアマッチAのトグル出力指定

４）ＴＧＲＡにパルス周期の１／２を設定

５）カウンタクリア（する必要はありませんが念のため）

６）ＴＰＵ＿３カウント開始指示

詳細は、前ページを参照して下さい。

６４～８３行：

ＰＢ［ＰＦ７］，ＰＢ［ＰＣ５］で出力周波数を指定させる操作をコントロールする関数です。

ここで、指定周波数のタイマ出力をさせ、その周波数の表示をしています。

2）汎用サブルーチン関係

file “CatSub.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプルプログラム＞                                                   */
  4: /*                                                                           */
  5: /*  ＜MOD＞       CatSub.c                                                   */
  6: /*  ＜役割＞      汎用ｻﾌﾞﾙｰﾁﾝ関係                                           */
  7: /*  ＜TAB＞       ４タブ編集                                                 */
  8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                             */
  9: /*                                                                           */
 10: /*****************************************************************************/
 11: #include    "io261x.h"
 12: #include    "DemoCtl.h"
 13: /*****************************************************************************/
 14: /*      ＢＩＮ－＞アスキーＤＥＣ変換 桁指定  asc[keta]=NULL 付き             */
 15: /*****************************************************************************/
 16: void    Bin2AdecN(char *asc,Ushort bin,Uchar keta)
 17: {
 18:     asc[keta] = 0;
 19:     while(keta--) {
 20:         asc[keta]  = bin % 10;
 21:         bin /= 10;
 22:         asc[keta] |= '0';
 23:     }
 24: }
 25: /*****************************************************************************/
 26: /*      ＢＩＮ－＞アスキーＨＥＸ変換 桁指定  asc[keta]=NULL 付き             */
 27: /*****************************************************************************/
 28: void    Bin2AhexN(char *asc,Ushort bin,Uchar keta)
 29: {
 30:     Uchar  dt;
 31:
 32:     asc[keta] = 0;
 33:     while(keta--) {
```

```
34:         dt = (bin & 0xf);
35:         if (dt >= 0xa) dt = (dt + 0x37);
36:         else           dt = (dt + 0x30);
37:         asc[keta] = dt;
38:         bin >>= 4;
39:     }
40: }
41: /*************************************************************************/
42: /*      アスキーＤＥＣ－＞ＢＩＮ変換 桁指定                              */
43: /*************************************************************************/
44: char *  Adec2binN(Ushort *bin,char *ptr,Uchar keta)
45: {
46:     char   dt;
47:
48:     *bin = 0;
49:     while(keta--) {
50:         dt   = (char)(*ptr - 0x30);
51:         *bin = (*bin * 10) + dt;
52:         ptr++;
53:     }
54:     return(ptr);
55: }
56: /*************************************************************************/
57: /*      アスキーＨＥＸ－＞ＢＩＮ変換 桁指定                              */
58: /*************************************************************************/
59: char *  Ahex2binN(Ushort *bin,char *ptr,Uchar keta)
60: {
61:     Uchar  dt;
62:     Uchar  c;
63:
64:     *bin  = 0;
65:     while(keta--) {
66:         c = (Uchar)toupper(*ptr);
67:         if (c >= 'A')  dt = c - 0x37;
68:         else           dt = c - 0x30;
69:         *bin = (*bin << 4) | dt;
```

```
 70:         ptr++;
 71:     }
 72:     return(ptr);
 73: }
 74: /*************************************************************************/
 75: /*      ストリングＣｏｐｙ(WORD)                                         */
 76: /*************************************************************************/
 77: Ushort * _strcpyW(Ushort *dst,Ushort *src)
 78: {
 79:     while(*src != 0) {
 80:         *dst++ = *src++;
 81:     }
 82:     *dst = 0;
 83:     return(dst);
 84: }
```

［リストの説明］

１６～２４行：

Ｕｓｈｏｒｔのバイナリを指定桁の１０進アスキー変換をする関数です。

２８～４０行：

Ｕｓｈｏｒｔのバイナリを指定桁の１６進アスキー変換をする関数です。

４４～５５行：

１０進アスキーデータの指定桁分をＵｓｈｏｒｔのバイナリに変換をする関数です。

５９～７３行：

１６進アスキーデータの指定桁分をＵｓｈｏｒｔのバイナリに変換をする関数です。

７７～８４行：

Ｗｏｒｄデータを元から先へゼロ（０）までコピーする関数です。

以後、この共通サブルーチンを各所で使用します。

3）メインコントロール

file “Cat261p5.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  ポーリング                                                 */
  4: /*                                                                           */
  5: /*  ＜MOD＞       Cat261p5.c                                                 */
  6: /*  ＜役割＞      main                                                       */
  7: /*  ＜TAB＞       ４タブ編集                                                 */
  8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                              */
  9: /*  ＜使用ﾊｰﾄﾞ＞ CAT-261-H8S/2612 エーワン(株)                                */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    "io261x.h"
 13: #include    "DemoCtl.h"
 14: /*****************************************************************************/
 15: /*      外部変数使用宣言                                                     */
 16: /*****************************************************************************/
 17: extern  short       BuzzerHz;          /* Buzzer Hz                         */
 18: /*****************************************************************************/
 19: /*      変数宣言                                                             */
 20: /*****************************************************************************/
 21:         Uchar       ModeStep;          /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                 */
 22:         Uchar       Shift;             /* shift ﾊﾟﾀｰﾝ                       */
 23: /*****************************************************************************/
 24: /*      _main()                                                              */
 25: /*****************************************************************************/
 26: void    _main(void)
 27: {
 28:     disable();                         /* PBC の割込の為                    */
 29:     SYSCR = 0x21;                      /* ｼｽﾃﾑｺﾝﾄﾛｰﾗ 割込ﾓｰﾄﾞ 2            */
 30:                                        /*             NMI 立下りエッジ      */
 31:                                        /*             RAME 有効             */
 32:     SoftWait1ms(400);                  /* 400msWait(ﾘｾｯﾄ遅延時間ﾊｰﾄﾞ)        */
 33:                                        /* H-ﾃﾞﾊﾞｯｶﾞ<->Target 通信可能に成るまで*/
```

```
34:                                     /* の１回リトライ時間分待つ(20回)    */
35:     MemInitial();                   /* ﾒﾓﾘ系初期化                        */
36:     IoInitial();                    /* I/O系初期化                        */
37:     while(1) {
38:         SigInput();                 /* Signal Input  Process              */
39:         SoftWait1ms(20);            /*        ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り      */
40:         ModeCntrol();               /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                         */
41:         AllLcdDisp();               /* LCD    全画面表示                  */
42:         SigOutput();                /* Signal Output Process(LED点灯)    */
43:     }
44: }
45: /*****************************************************************************/
46: /*      Ｍｅｍ初期化                                                         */
47: /*****************************************************************************/
48: void    MemInitial(void)
49: {
50:     ModeStep = 0;                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                  */
51:     Shift = 0;                      /* Led Disp Patan Initial             */
52:
53:     PioMemInitial();                /* PIO    Mem 初期化                  */
54:     LcdMemInitial();                /* LCD    Mem 初期化                  */
55:     TimMemInitial();                /* TIM    Mem 初期化                  */
56: }
57: /*****************************************************************************/
58: /*      Ｉ／Ｏ初期化                                                         */
59: /*****************************************************************************/
60: void    IoInitial(void)
61: {
62:     PioIoInitial();                 /* PIO    I/O 初期化                  */
63:     LcdIoInitial();                 /* LCD    I/O 初期化                  */
64:     TimIoInitial();                 /* TIM    I/O 初期化                  */
65: }
66: /*****************************************************************************/
67: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                          */
68: /*****************************************************************************/
69: void    ModeCntrol()
```

```
70: {
71:     if (GetUpPort(1) & 0x1) {           /* PB[PA0] ON?(立上)                  */
72:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST     */
73:         else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST     */
74:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ      */
75:         Buzzer(0);                                  /*  強制 OFF            */
76:     }
77:     switch(ModeStep) {
78:     case 0:
79:         GotoxyMemSet(0,0,"CAT261&AH6000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ       */
80:         GotoxyMemSet(0,1,"Polling    [PA0]");
81:         ModeStep++;
82:         break;
83:     case 1:
84:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ       */
85:         break;
86:     case 10:                            /*  PIO   TEST                        */
87:         GotoxyMemSet(0,0,"PIO             ");
88:         GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
89:         ModeStep++;
90:         break;
91:     case 11:
92:         PioDemo();
93:         break;
94:     case 20:                            /*  Timer TEST                        */
95:         GotoxyMemSet(0,0,"Timer/Counter   ");
96:         GotoxyMemSet(0,1,"0000Hz[+PF7-PC5]");
97:         BuzzerHz = 0;                   /* Buzzer Hz                          */
98:         ModeStep++;
99:         break;
100:    case 21:
101:        TimerDemo();
102:        RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ       */
103:        break;
104:    }
105: }
```

```
106: /*************************************************************************/
107: /*      RunRun()        CPU 走行表示                                     */
108: /*************************************************************************/
109: void    RunRun()
110: {
111:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
112:     PutOutPort(Shift,'=');
113: }
114: /*************************************************************************/
115: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                 */
116: /*************************************************************************/
117: void    SoftWait1ms(Ushort ms)
118: {
119:     while(ms-- != 0) {
120:         Wait1ms();
121:     }
122: }
123: /*************************************************************************/
124: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                               */
125: /*************************************************************************/
126: void    SoftWait10us(Ushort us)
127: {
128:     while(us-- != 0) {
129:         Wait10us();
130:     }
131: }
132: /*************************************************************************/
133: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait               */
134: /*************************************************************************/
135: void    Wait1ms()
136: {
137:     asm("       push.w  r0                  ");
138:     asm("       mov.w   #5000,r0            ");     /* 5000*4=20000cyc  */
139:     asm(" wait:                             ");
140:     asm("       dec.w   #1,r0               ");     /* 1 clock          */
141:     asm("       bne     wait:16             ");     /* 3 clock          */
```

```
142:     asm("       pop.w   r0                      ");
143: }
144: /*************************************************************************/
145: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                */
146: /*************************************************************************/
147: void    Wait10us()
148: {
149:     asm("       push.w  r0                      ");
150:     asm("       mov.w   #50,r0                  ");     /* 50*4=200cyc      */
151:     asm(" wait1:                                ");
152:     asm("       dec.w   #1,r0                   ");     /* 1 clock          */
153:     asm("       bne     wait1:16                ");     /* 3 clock          */
154:     asm("       pop.w   r0                      ");
155: }
```

［リストの説明］　前章のメインコントロールから追加された部分だけ解説します。

１７行：

このモジュールで使用する外部変数宣言です。

５５行：

“ＰｏｌＴｉｍｅ．ｃ”で使用する変数の初期化関数を呼んでいます。

６４行：

タイマレジスタを初期化する関数を呼んでいます。

９４～１０３行：

この章のサンプルプログラムを動作させるための制御部分を追加しました。

これで、この章のリスト説明は終わりです。

ご理解いただけたでしょうか？

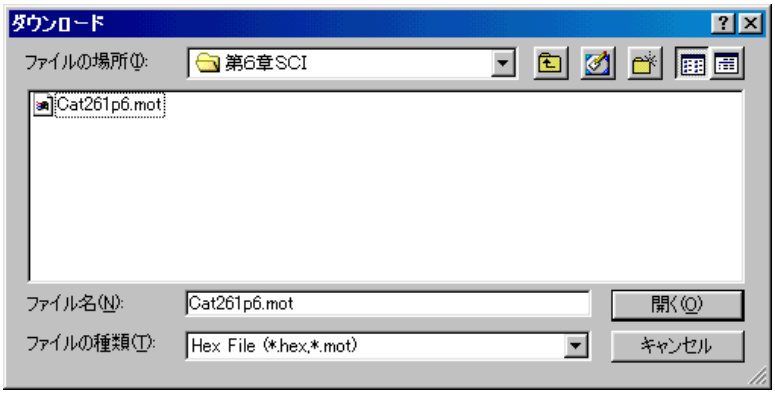
周波数を１００Ｈｚごとの変化でなく、もっと細かくしたい場合は、どこを修正すれば良いか、わかりましたか？（ＰｏｌＴｉｍｅ．ｃのどこか）

興味のあるかたは、修正して“ｍａｋｅ”を実行してチャレンジしてみて下さい。

次は、ＳＣＩ使用例の解説へと進みます。

第6章　ＳＣＩ

　ここの章では、ＳＣＩのイニシャルとＳＣＩ使用サンプルの解説をします。
ＳＣＩ使用サンプルは、ポーリング送／受信の１バイトのループバック通信です。
ＳＣＩ０（送信）―＞ＳＣＩ１（受信）
ＳＣＩ１（送信）―＞ＳＣＩ０（受信）　の方式でループバックします。
まずは、ＤＥＦのダウンロードで、



¥評価ボード¥第１部ポーリング編¥第６章ＳＣＩ¥
にディレクトリの移動をしておいて下さい。

１．動かしてみましょう

　移動したディレクトリの中に、“Ｃａｔ２６１ｐ６．ｍｏｔ”というＨＥＸファイルがあります。
これをダウンロードしてから、プログラム実行してみて下さい。

どうです？　ＬＣＤにオープニングメッセージがでましたか？

もう馴れたと思いますので、**ＬＣＤ表示の左上に“ＳＣＩ”と表示がでるまで、**
ＰＢ［ＰＡ０］を押して下さい。
［評価ボード上の**ＳＷ１１－１，２をオン**（ＴＸＤ，ＲＸＤを折り返す）にして下さい。］
［ＣＡＴ２６１基板の**ＳＷ１－１，２，３，４をオン**（信号をＲＳドライバ経由）にして下さい。］

ＰＢ［ＰＣ２］が、送受信開始／停止になっていますので、押してみて下さい。
これで、ＰＢ［ＰＣ２］停止を押されるまで、送受信を繰り返します。

ＬＣＤ画面　ＴｘｘＲｏｘｘＲｉｘｘ［ＰＣ２］　Ｒｏ＝ＳＣＩ０　Ｒｉ＝ＳＣＩ１の受信
と“ｘｘ”の部分は、送信するごとに＋１する送受信データです。
送受信停止中に、ＰＢ［ＰＦ７］、ＰＢ［ＰＣ５］を押しますと、ボーレートの変更ができます。
それでは、どのような仕組みでプログラムされているかプログラムリストを見てみましょう！

２．プログラムリスト

このサンプルは、前章に１モジュール追加して、８ファイルの構成になっています。

“ｖｅｃｔｏｒＡ．ａｓｍ”	前章のまま使用したので解説を省略します。
“Ｓｔａｒｔｕｐ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＰｉｏ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＬｃｄ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＴｉｍｅ．ｃ”	前章のまま使用したので解説を省略します。
“ＣａｔＳｕｂ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＳｃｉ．ｃ”	ＳＣＩコントロールのモジュールです。
“Ｃａｔ２６１ｐ５．ｃ”	メインコントロール部です。

１）ＳＣＩコントロール関係

file “PolSci.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  ポーリング                                                 */
  4: /*                                                                           */
  5: /*  ＜MOD＞       PolSci.c                                                   */
  6: /*  ＜役割＞      SCI(SIO) 関係                                              */
  7: /*  ＜TAB＞       ４タブ編集                                                 */
  8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                             */
  9: /*  ＜使用ﾊｰﾄﾞ＞  CAT-261-H8S/2612 エーワン(株)                              */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <string.h>
 13: #include    "io261x.h"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*      SCI ボレート計算+etc                                                 */
 17: /*****************************************************************************/
 18: #define CLOCK       (20000000/32)       /* 調歩同期 n=0 時の計算式          */
 19: #define BRRN(b)     ((CLOCK/b)-1)
 20: #define NON         0                   /* ﾊﾟﾘﾃｨ     NON                   */
 21: #define EVEN        1                   /*           EVEN                  */
```

```
#define ODD         2                   /*          ODD                     */
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       SciStep;            /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
        Uchar       SciSelect;          /* ﾎﾞｰﾚｰﾄ選択                       */
        Uchar       TxDat;              /* 送信ﾃﾞｰﾀ                         */
        Uchar       RxDat;              /* 受信ﾃﾞｰﾀ                         */
const   Ushort      BpsTbl[] =
{
        4800,
        9600,
        19200,
        31250,
        38400,
        0,
};
/*****************************************************************************/
/*      Ｍｅｍ初期化                                                         */
/*****************************************************************************/
void    SciMemInitial(void)
{
    SciStep   = 0;                      /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
    SciSelect = 1;                      /* 9600BPS                          */
}
/*****************************************************************************/
/*      Ｉ／Ｏ初期化                                                         */
/*****************************************************************************/
void    SciIoInitial(void)
{
    RsOpen0(9600,8,NON);                /* SCI0 Open                        */
    RsOpen1(9600,8,NON);                /* SCI1 Open                        */
}
/*****************************************************************************/
/*  RS232C(SCI0) Open bps    = ﾎﾞﾚｰﾄ[4800,*9600,19200,31250,38400]          */
/*                     ch    = ｷｬﾗｸﾀ[7,*8]                                   */
```

```
/*                    pty     = ﾊﾟﾘﾃｨ[*0=NON 1=EVEN 2=ODD]                  */
/*                    *       = ﾃﾞﾌｫﾙﾄ                                       */
/*                    固定    = x1 STOP=1bit                                 */
/*****************************************************************************/
void    RsOpen0(Ushort bps,Uchar ch,Uchar pty)
{
    Uchar   bck;
    Uchar   smr;

    MSTPCRB &= ~0x80;                   /* ﾓｼﾞｭｰﾙｽﾄｯﾌﾟ SCI0 ｽﾀｰﾄ            */
    SCMR_0 = 0xf2;                      /* ｽﾏｰﾄ SMIF=0 Non ｽﾏｰﾄ              */

    smr = 0;                            /* ﾓｰﾄﾞ ｷｬﾗｸﾀ[8bit] ﾊﾟﾘﾃｨ[NON]      */
    if (ch == 7)          smr |= 0x40;  /*      ｷｬﾗｸﾀ[7bit]                 */
    if (pty == EVEN)      smr |= 0x20;  /*      ﾊﾟﾘﾃｨ[EVEN]                 */
    else if (pty == ODD) smr |= 0x30;   /*      ﾊﾟﾘﾃｨ[ODD]                  */
    SMR_0 = smr;

    if (bps == 4800)        bck = BRRN(4800);
    else if (bps == 9600)   bck = BRRN(9600);
    else if (bps == 19200)  bck = BRRN(19200);
    else if (bps == 31250)  bck = BRRN(31250);
    else if (bps == 38400)  bck = BRRN(38400);
    BRR_0 = bck;                        /* ﾎﾞｰﾚｰﾄｼﾞｪﾈﾚｰﾄ設定              */
    Wait1ms();                          /* 1bit 経過待ち 2400BPS(0.42ms+a)  */
    SSR_0 &= 0x87;                      /*  RDRF+ResetStatus               */
    SCR_0 = 0x30;                       /*  TE+RE                          */
}
/*****************************************************************************/
/*  RS232C(SCI1) Open bps    = ﾎﾞﾚｰﾄ[4800,*9600,19200,31250,38400]          */
/*                    ch      = ｷｬﾗｸﾀ[7,*8]                                  */
/*                    pty     = ﾊﾟﾘﾃｨ[*0=NON 1=EVEN 2=ODD]                  */
/*                    *       = ﾃﾞﾌｫﾙﾄ                                       */
/*                    固定    = x1 STOP=1bit                                 */
/*****************************************************************************/
void    RsOpen1(Ushort bps,Uchar ch,Uchar pty)
```

```
 94: {
 95:     Uchar   bck;
 96:     Uchar   smr;
 97:
 98:     MSTPCRB &= ~0x40;                   /* ﾓｼﾞｭｰﾙｽﾄｯﾌﾟ SCI1 ｽﾀｰﾄ            */
 99:     SCMR_1 = 0xf2;                      /* ｽﾏｰﾄ SMIF=0 Non ｽﾏｰﾄ              */
100:
101:     smr = 0;                           /* ﾓｰﾄﾞ ｷｬﾗｸﾀ[8bit] ﾊﾟﾘﾃｨ[NON]      */
102:     if (ch == 7)          smr |= 0x40;  /*       ｷｬﾗｸﾀ[7bit]                */
103:     if (pty == EVEN)      smr |= 0x20;  /*       ﾊﾟﾘﾃｨ[EVEN]                */
104:     else if (pty == ODD) smr |= 0x30;   /*       ﾊﾟﾘﾃｨ[ODD]                 */
105:     SMR_1 = smr;
106:
107:     if (bps == 4800)         bck = BRRN(4800);
108:     else if (bps == 9600)    bck = BRRN(9600);
109:     else if (bps == 19200)   bck = BRRN(19200);
110:     else if (bps == 31250)   bck = BRRN(31250);
111:     else if (bps == 38400)   bck = BRRN(38400);
112:     BRR_1 = bck;                       /* ﾎﾞｰﾚｰﾄｼﾞｪﾈﾚｰﾄ設定              */
113:     Wait1ms();                         /* 1bit 経過待ち 2400BPS(0.42ms+a)  */
114:     SSR_1 &= 0x87;                     /* RDRF+ResetStatus                 */
115:     SCR_1 = 0x30;                      /* TE+RE                            */
116: }
117: /*************************************************************************/
118: /*      RsPutch0    RS232C(SCI0) 送信                                    */
119: /*************************************************************************/
120: void    RsPutch0(Uchar tx)
121: {
122:     while((SSR_0 & 0x80) == 0) {}           // TDRE=1 wait
123:     TDR_0 = tx;
124:     SSR_0 &= ~(0x80);
125: }
126: /*************************************************************************/
127: /*      RsPutch1    RS232C(SCI1) 送信                                    */
128: /*************************************************************************/
129: void    RsPutch1(Uchar tx)
```

```
130: {
131:     while((SSR_1 & 0x80) == 0) {}           // TDRE=1 wait
132:     TDR_1 = tx;
133:     SSR_1 &= ~(0x80);
134: }
135: /*****************************************************************************/
136: /*      RsGetch0    RS232C(SCI0) 受信                                        */
137: /*****************************************************************************/
138: short    RsGetch0()
139: {
140:     Ushort  time;
141:     Uchar   dt;
142:
143:     time = 0;
144:     while((SSR_0 & 0x78) == 0) {        /* RDRF+OE+FE+PE   ON ?            */
145:         SoftWait1ms(1);                 /* 1ms                             */
146:         if (++time >= 20) return(-1);   /* Error                           */
147:     }
148:     dt = RDR_0;                         /* 受信                            */
149:     if (SSR_0 & 0x38) {                 /* SCI  OE+FE+PE Error             */
150:         SSR_0 &= 0x87;                  /* Error Reset                     */
151:         return(-1);                     /* Error Return                    */
152:     }
153:     SSR_0 &= ~(0x40);                   /* RDRF OFF                        */
154:     return(dt);
155: }
156: /*****************************************************************************/
157: /*      RsGetch1    RS232C(SCI1) 受信                                        */
158: /*****************************************************************************/
159: short    RsGetch1()
160: {
161:     Ushort  time;
162:     Uchar   dt;
163:
164:     time = 0;
165:     while((SSR_1 & 0x78) == 0) {        /* RDRF+OE+FE+PE   ON ?            */
```

```
166:         SoftWait1ms(1);                 /* 1ms                              */
167:         if (++time >= 20) return(-1);   /* Error                            */
168:     }
169:     dt = RDR_1;                         /* 受信                             */
170:     if (SSR_1 & 0x38) {                 /* SCI  OE+FE+PE Error              */
171:         SSR_1 &= 0x87;                  /* Error Reset                      */
172:         return(-1);                     /* Error Return                     */
173:     }
174:     SSR_1 &= ~(0x40);                   /* RDRF OFF                         */
175:     return(dt);
176: }
177: /*****************************************************************************/
178: /*      SciDemo()      ＵＳＡＲＴデモ                                        */
179: /*****************************************************************************/
180: void    SciDemo()
181: {
182:     Uchar   dec[2+1];
183:     short   stat;
184:
185:     SciSequence();                      /* SCI 操作ｺﾝﾄﾛｰﾙ                    */
186:     switch(SciStep) {
187:     case 0:
188:         break;
189:     case 1:
190:         RsOpen0(BpsTbl[SciSelect],8,NON);   /* RS232C(SCI0) Open            */
191:         RsOpen1(BpsTbl[SciSelect],8,NON);   /* RS232C(SCI1) Open            */
192:         GotoxyMemSet(0,0,"TxxRoxxRixx");
193:         TxDat = 0;
194:         SciStep++;
195:         break;
196:     case 2:
197:         Bin2AhexN(dec,TxDat,2);         /* 表示用送信データ作成(SCI0)       */
198:         GotoxyMemSet(1,0,dec);
199:         RsPutch0(TxDat);                /* 送信(SCI0)                       */
200:         SciStep++;
201:         break;
```

```
202:     case 3:
203:         stat = RsGetch1();
204:         if (stat == -1) strcpy(dec,"ee");      /* Error                        */
205:         else            Bin2AhexN(dec,stat,2);/* 表示用受信データ作成         */
206:         GotoxyMemSet(9,0,dec);
207:         SciStep++;
208:         break;
209:     case 4:
210:         RsPutch1(TxDat++);              /* 送信(SCI1)                       */
211:         SciStep++;
212:         break;
213:     case 5:
214:         stat = RsGetch0();
215:         if (stat == -1) strcpy(dec,"ee");      /* Error                        */
216:         else            Bin2AhexN(dec,stat,2);/* 表示用受信データ作成         */
217:         GotoxyMemSet(5,0,dec);
218:         SciStep = 2;
219:         break;
220:     }
221: }
222: /*****************************************************************************/
223: /*      SciSequence()   Usart 操作ｺﾝﾄﾛｰﾙ                                     */
224: /*****************************************************************************/
225: void    SciSequence()
226: {
227:     Uchar   port;
228:     Uchar   dec[5+1];
229:
230:     port = GetUpPort(1);                /* PB[PA0]->PB[PF7]                 */
231:     if (port & 0xe) {                   /* PB[P31]->PB[P33] ON(立上) ?      */
232:         if (port & 0x2) {               /* PB[PC2] ON ? 送信ｽﾀｰﾄ/ｽﾄｯﾌﾟ      */
233:             if (SciStep == 0) SciStep = 1;
234:             else              SciStep = 0;
235:         }
236:         if (SciStep == 0) {             /* 停止中のみ受け付ける             */
237:             if (port & 0x4) {           /* PB[PC5] ON ? ﾎﾞﾚｰﾄ下げ?          */
```

```
238:                 if (SciSelect != 0) --SciSelect;
239:             }
240:             else if (port & 0x8) {      /* PB[PF7] ON ? ﾎﾞﾚｰﾄ上げ?          */
241:                 if (BpsTbl[SciSelect+1] != 0) ++SciSelect;
242:             }
243:             Bin2AdecN(dec,BpsTbl[SciSelect],5); /* 表示用データ作成          */
244:             GotoxyMemSet(0,1,dec);
245:         }
246:     }
247: }
```

リストの説明に入る前に、ＳＣＩ関係資料を添付します。

チャネル	端子名	入出力	機能
0	ＲＸＤ０ ＴＸＤ０	入力 出力	チャネル０の受信データ入力端子 チャネル０の送信データ出力端子
1	ＲＸＤ１ ＴＸＤ１	入力 出力	チャネル１の受信データ入力端子 チャネル１の送信データ出力端子

チャネル	名　称	シンボル	アドレス
0	シリアルモードレジスタ０ ビットレートレジスタ０ シリアルコントロールレジスタ０ トランスミットデータレジスタ０ シルアルステータスレジスタ０ レシーブデータレジスタ０	ＳＭＲ＿０ ＢＲＲ＿０ ＳＣＲ＿０ ＴＤＲ＿０ ＳＳＲ＿０ ＲＤＲ＿０	０ｘｆｆｆｆ７８ ０ｘｆｆｆｆ７９ ０ｘｆｆｆｆ７ａ ０ｘｆｆｆｆ７ｂ ０ｘｆｆｆｆ７ｃ ０ｘｆｆｆｆ７ｄ
1	シリアルモードレジスタ１ ビットレートレジスタ１ シリアルコントロールレジスタ１ トランスミットデータレジスタ１ シルアルステータスレジスタ１ レシーブデータレジスタ１	ＳＭＲ＿１ ＢＲＲ＿１ ＳＣＲ＿１ ＴＤＲ＿１ ＳＳＲ＿１ ＲＤＲ＿１	０ｘｆｆｆｆ８０ ０ｘｆｆｆｆ８１ ０ｘｆｆｆｆ８２ ０ｘｆｆｆｆ８３ ０ｘｆｆｆｆ８４ ０ｘｆｆｆｆ８５

表［６－２－１］　ＳＣＩの端子名とＩ／Ｏマップ

シリアルモードレジスタ（ＳＭＲ＿ｘ）				
ﾋﾞｯﾄ	名前	初期値	R/W	機能
7	C/A	0	R/W	０：調歩同期モード １：クロック同期式モード
6	CHR	0	R/W	０：データ長８ビット　クロック同期では８ビット固定 １：データ長７ビット　LSB ファースト固定
5	PE	0	R/W	０：パリティディセーブル １：パリティイネーブル
4	O/E	0	R/W	０：偶数パリティ (EVEN) １：奇数パリティ (ODD)
3	STOP	0	R/W	０：１ストップビット １：２ストップビット
2	MP	0	R/W	０：マルチプロセッサ通信ディセーブル １：マルチプロセッサ通信イネーブル
1 0	CKS1 CKS0	0 0	R/W	内蔵ボーレートジェネレートのクロックソース選択 ００：φクロック ０１：φ／４クロック １０：φ／１６クロック １１：φ／６４クロック

表［６－２－２］　シリアルモードレジスタ

シリアルコントロールレジスタ（ＳＣＲ＿ｘ）				
ﾋﾞｯﾄ	名前	初期値	R/W	機能
7	TIE	0	R/W	０：ＴＸＩ割り込み要求ディセーブル １：ＴＸＩ割り込み要求イネーブル
6	RIE	0	R/W	０：ＲＸＩ／ＥＲＩ割り込み要求ディセーブル １：ＲＸＩ／ＥＲＩ割り込み要求イネーブル
5	TE	0	R/W	０：送信動作不可 １：送信動作可能
4	RE	0	R/W	０：受信動作不可 １：受信動作可能
3	MPIE	0	R/W	マルチプロセッサインタラプトイネーブル
2	TEIE	0	R/W	０：ＴＥＩ割り込み要求ディセーブル １：ＴＥＩ割り込み要求イネーブル
1 0	CKE1 CKE0	0 0	R/W	クロックソースの選択 調歩同期の場合　００：内部クロック ０１：内部クロック（ＳＣＫからビットレート出力） １ｘ：外部クロック（ＳＣＫへ入力）

表［６－２－３］　シリアルコントロールレジスタ

シリアルステータスレジスタ（ＳＳＲ＿ｘ）				
ﾋﾞｯﾄ	名前	初期値	R/W	機能
7	TDRE	1	R/W	１：ＴＤＲからＴＳＲにデータ転送された時
6	RDRF	0	R/W	１：受信正常終了　ＲＳＲからＲＤＲへ受信データ転送完了
5	ORER	0	R/W	１：オーバランエラー発生
4	FER	0	R/W	１：フレミングエラー発生
3	PER	0	R/W	１：パリティエラー発生
2	TEND	1	R	１：送信データの最終ビットが送信された時
1	MPB	0	R	マルチプロセッサビット
0	MPBT	0	R/W	マルチプロセッサビットトランスファ

表［６－２－４］　シリアルステータスレジスタ

［リストの説明］

１８～１９行：

調歩同期式モードでの通信時、指定ボーレートを得るためにビットレートレジスタへ与える数値を求める計算式の定義です。

ＢＲＲ値＝（(システムクロック／３２）／ボーレート）－１

［ＳＭＲへの内蔵ボーレートジェネレータのクロックソースをゼロ（φクロック）とする］

［システムクロック＝２０．０ＭＨｚ］

［ボーレート＝４８００，９６００，１９２００，３１２５０，３８４００］

２６行：

送受信をコントロールする変数宣言です。

２７行：

ボーレートを選択する変数宣言です。

２８行：

送信データを記憶する変数宣言です。

２９行：

受信データを記憶する変数宣言です。

３０～３８行：

選択できるボーレートテーブルの変数宣言です。

４２～４６行：

このモジュールで使用する変数の初期化関数です。

メインのメモリ初期化の時に呼ばれます。

５０～５４行：

ＳＣＩ０を初期化する関数“ＲｓＯｐｅｎ０”を呼んでいます。

ＳＣＩ１を初期化する関数“ＲｓＯｐｅｎ１”を呼んでいます。

デフォルトで、９６００ｂｐｓ、８ビット、パリティＮＯＮの仕様にしています。

メインのＩ／Ｏ初期化の時に呼ばれます。

６２～８５行：

ＳＣＩ０を初期化する関数です。

重要　モジュールストップコントロールレジスタのＳＣＩ０をオフ（開始）することをお忘れなく！

次にこの関数は、３個の引数を持っています。

第１引数は、ボーレート［２４００，４８００，９６００，１９２００，３８４００］

第２引数は、キャラクタ長［７，８］

第３引数は、パリティ［０＝ＮＯＮ（ディセーブル）　１＝偶数　２＝基数］

ストップビットは、１ビット固定とします。

［６７行］

モジュールストップコントロールレジスタのＳＣＩ０をオフ（開始）しています。

［６８行］

スマートカードインタフェースをディセーブルにしています。

［７０～７４行］

シリアルモードレジスタを設定するために、通信フォームの引数を分析して設定データを作成し設定しています。

［７６～８１行］

ビットレートレジスタを設定するために、調歩同期式モードでの通信ボーレートを得るための数値を求め設定しています。

［８２行］

１ビット送信経過時間以上待っています。

［８３行］

シリアルステータスレジスタの“ＲＤＲＦ”＋“ＯＲＥＲ”＋“ＦＥＲ”＋“ＰＥＲ”をリセットしています。

［８４行］

シリアルコントロールレジスタの“ＴＥ”＋“ＲＥ”をイネーブルにしています。

９３～１１６行：

ＳＣＩ１を初期化する関数です。

重要　モジュールストップコントロールレジスタのＳＣＩ１をオフ（開始）することをお忘れなく！

ＳＣＩ０の初期化と同じですので省略します。

１２０～１２５行：

ＳＣＩ０の１バイトデータを送信する関数です。

１２９～１３４行：

ＳＣＩ１の１バイトデータを送信する関数です。

１３８～１５５行：

ＳＣＩ０の１バイトデータを受信する関数です。

ＲＤＲＦ（受信あり）か、受信エラーが発生するまで待つループには、２０ｍｓのソフトタイマーが入れてあります。

受信エラーが発生した場合は、エラーリセットコマンドを発行後、ｓｈｏｒｔ（－１）を返しています。

正常受信した場合は、受信データを呼び先へ返します。

１５９～１７６行：

ＳＣＩ１の１バイトデータを受信する関数です。

ＳＣＩ０の１バイトデータを受信と同じですので省略します。

１８０～２２１行：

ＳＣＩ０とＳＣＩ１の動作確認をするための関数です。

ＳＣＩ０／１の設定、１バイト送信、１バイト受信、送受信データの表示等を処理しています。

ＳＣＩ０（1バイト送信）　――――＞　ＳＣＩ１（1バイト受信）

ＳＣＩ１（1バイト送信）　――――＞　ＳＣＩ０（1バイト受信）

の繰り返しをしています。

２２５～２４７行：

ＳＣＩの動作確認をする場合、ＰＢ操作のコントロールを管理する関数です。

2）メインコントロール

file “Cat261p6.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  ポーリング                                                 */
  4: /*                                                                           */
  5: /*  ＜MOD＞       Cat261p6.c                                                 */
  6: /*  ＜役割＞      main                                                       */
  7: /*  ＜TAB＞       ４タブ編集                                                 */
  8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                              */
  9: /*  ＜使用ﾊｰﾄﾞ＞ CAT-261-H8S/2612 エーワン(株)                                */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    "io261x.h"
 13: #include    "DemoCtl.h"
 14: /*****************************************************************************/
 15: /*      外部変数使用宣言                                                     */
 16: /*****************************************************************************/
 17: extern  short       BuzzerHz;          /* Buzzer Hz                        */
 18: extern  Uchar       SciStep;           /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
 19: extern  Uchar       SciSelect;         /* ﾎﾞｰﾚｰﾄ選択                        */
 20: /*****************************************************************************/
 21: /*      変数宣言                                                             */
 22: /*****************************************************************************/
 23:         Uchar       ModeStep;          /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
 24:         Uchar       Shift;             /* shift ﾊﾟﾀｰﾝ                       */
 25: /*****************************************************************************/
 26: /*      _main()                                                              */
 27: /*****************************************************************************/
 28: void    _main(void)  {
 29:     disable();                         /* PBC の割込の為                   */
 30:     SYSCR = 0x21;                      /* ｼｽﾃﾑｺﾝﾄﾛｰﾗ 割込ﾓｰﾄﾞ 2           */
 31:                                        /*             NMI 立下りエッジ     */
 32:                                        /*             RAME 有効            */
 33:     SoftWait1ms(400);                  /* 400msWait(ﾘｾｯﾄ遅延時間ﾊｰﾄﾞ)       */
```

```
34:                                     /* H-ﾃﾞﾊﾞｯｶﾞ<->Target 通信可能に成るまで*/
35:                                     /* の１回リトライ時間分待つ(20 回)     */
36:     MemInitial();                   /* ﾒﾓﾘ系初期化                          */
37:     IoInitial();                    /* I/O 系初期化                         */
38:     while(1) {
39:         SigInput();                 /* Signal Input  Process               */
40:         SoftWait1ms(20);            /*        ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り       */
41:         ModeCntrol();               /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                           */
42:         AllLcdDisp();               /* LCD    全画面表示                    */
43:         SigOutput();                /* Signal Output Process(LED 点灯)     */
44:     }
45: }
46: /*************************************************************************/
47: /*      Ｍｅｍ初期化                                                     */
48: /*************************************************************************/
49: void    MemInitial(void)
50: {
51:     ModeStep = 0;                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                    */
52:     Shift = 0;                      /* Led Disp Patan Initial              */
53:
54:     PioMemInitial();                /* PIO    Mem 初期化                   */
55:     LcdMemInitial();                /* LCD    Mem 初期化                   */
56:     TimMemInitial();                /* TIM    Mem 初期化                   */
57:     SciMemInitial();                /* SCI    Mem 初期化                   */
58: }
59: /*************************************************************************/
60: /*      Ｉ／Ｏ初期化                                                     */
61: /*************************************************************************/
62: void    IoInitial(void)
63: {
64:     PioIoInitial();                 /* PIO    I/O 初期化                   */
65:     LcdIoInitial();                 /* LCD    I/O 初期化                   */
66:     TimIoInitial();                 /* TIM    I/O 初期化                   */
67:     SciIoInitial();                 /* SCI    I/O 初期化                   */
68: }
69: /*************************************************************************/
```

```
/*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                              */
/*************************************************************************/
void    ModeCntrol()
{
    if (GetUpPort(1) & 0x1) {           /* PB[PA0] ON?(立上)                 */
        if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST     */
        else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST     */
        else if (ModeStep < 30) ModeStep = 30;      /*  SCI   Goto TEST     */
        else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ      */
        Buzzer(0);                                  /*  強制 OFF            */
    }
    switch(ModeStep) {
    case 0:
        GotoxyMemSet(0,0,"CAT261&AH6000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ       */
        GotoxyMemSet(0,1,"Polling    [PA0]");
        ModeStep++;
        break;
    case 1:
        RunRun();                       /* ｼﾌﾄLED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ       */
        break;
    case 10:                            /*  PIO   TEST                      */
        GotoxyMemSet(0,0,"PIO           ");
        GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
        ModeStep++;
        break;
    case 11:
        PioDemo();
        break;
    case 20:                            /*  Timer TEST                      */
        GotoxyMemSet(0,0,"Timer/Counter  ");
        GotoxyMemSet(0,1,"0000Hz[+PF7-PC5]");
        BuzzerHz = 0;                   /* Buzzer Hz                        */
        ModeStep++;
        break;
    case 21:
        TimerDemo();
```

```
106:        RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ        */
107:        break;
108:    case 30:                            /* SCI TEST                          */
109:        GotoxyMemSet(0,0,"SCI   8,n,1[PC2]");
110:        GotoxyMemSet(0,1,"09600b[+PF7-PC5]");
111:        SciStep   = 0;                  /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                       */
112:        SciSelect = 1;                  /* 9600BPS                           */
113:        ModeStep++;
114:        break;
115:    case 31:
116:        SciDemo();
117:        RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ        */
118:        break;
119:    }
120: }
121: /*****************************************************************************/
122: /*      RunRun()        CPU 走行表示                                         */
123: /*****************************************************************************/
124: void    RunRun()
125: {
126:    if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
127:    PutOutPort(Shift,'=');
128: }
129: /*****************************************************************************/
130: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                    */
131: /*****************************************************************************/
132: void    SoftWait1ms(Ushort ms)
133: {
134:    while(ms-- != 0) {
135:        Wait1ms();
136:    }
137: }
138: /*****************************************************************************/
139: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                   */
140: /*****************************************************************************/
141: void    SoftWait10us(Ushort us)
```

```
142: {
143:     while(us-- != 0) {
144:         Wait10us();
145:     }
146: }
147: /*****************************************************************************/
148: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                    */
149: /*****************************************************************************/
150: void    Wait1ms()
151: {
152:     asm("       push.w  r0                      ");
153:     asm("       mov.w   #5000,r0                ");      /* 5000*4=20000cyc  */
154:     asm(" wait:                                 ");
155:     asm("       dec.w   #1,r0                   ");      /* 1 clock          */
156:     asm("       bne     wait:16                 ");      /* 3 clock          */
157:     asm("       pop.w   r0                      ");
158: }
159: /*****************************************************************************/
160: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                   */
161: /*****************************************************************************/
162: void    Wait10us()
163: {
164:     asm("       push.w  r0                      ");
165:     asm("       mov.w   #50,r0                  ");      /* 50*4=200cyc      */
166:     asm(" wait1:                                ");
167:     asm("       dec.w   #1,r0                   ");      /* 1 clock          */
168:     asm("       bne     wait1:16                ");      /* 3 clock          */
169:     asm("       pop.w   r0                      ");
170: }
```

［リストの説明］　前章のメインコントロールから追加された部分だけ解説します。

１８～１９行：

このモジュールで使用する外部変数宣言を追加しました。

５７行：

“ＰｏｌＳｃｉ．ｃ”で使用する変数の初期化関数を呼んでいます。

６７行：

ＳＣＩを初期化する関数を呼んでいます。

１０８～１１８行：

この章のサンプルプログラムを動作させるための制御部分を追加しました。

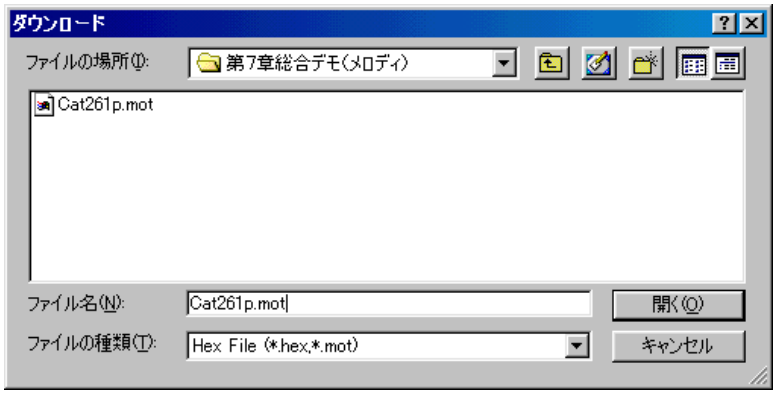
以上で、この章の説明は終わりにします。

次は、ポーリングにおける総合デモ（メロディ）の解説へと進みます。

第７章　総合デモ（メロディ）

ここの章では、ポーリングでの総合デモ（メロディ）です。
評価ボードに付いているブザーを使い、チョットした演奏をします。

まずは、ＤＥＦのダウンロードで、



￥評価ボード￥第１部ポーリング編￥第７章総合デモ（メロディ）￥
にディレクトリの移動をしておいて下さい。

１．動かしてみましょう

移動したディレクトリの中に、**“Ｃａｔ２６１ｐ．ＨＥＸ”**というＨＥＸファイルがあります。
これをダウンロードしてから、プログラム実行してみて下さい。

どうです？　ＬＣＤにオープニングメッセージがでましたか？

もう馴れたと思いますので、**ＬＣＤ表示の左上に“Ｍｅｌｏｄｙ”と表示がでるまで、**
ＰＢ［ＰＡ０］を押して下さい。
ＬＣＤの下行　ＰＣ２［Ｍ］Ｆ７［ス］Ｃ５［セ］　と表示しているはずです。
ＰＣ２［Ｍ］は、マニュアルの意味です。

ＳＷ［Ｐ４０］－＞ＳＷＰ［４７］を、オン／オフしてみて下さい。
ド、レ、ミ……と音がするはずです。

ＰＢ［ＰＣ２］を押して下さい。（モード変更）
ＰＣ２［Ａ］と表示したはずです。（Ａｕｔｏ演奏の意味）

ＰＢ［ＰＣ５］を押して下さい。（選曲）
　ネコフンジャッター＞イヌノオマワリサン－＞アマリリス（好きな曲で止めて下さい）

ＰＢ［ＰＦ７］を押して下さい（開始／停止）
　演奏したはずです。（音痴ですみません）

止めたい時は、どれかＰＢを長く押して下さい。
演奏の音の長さは、ポーリング記述のためソフトタイマを使用しています。
ソフトタイマ使用中は、他の処理は完全に停止してしまうため、ＰＢを長く押す必要があるわけです。
また、演奏中ＬＥＤが止まって見えるはずです、これもソフトタイマの影響です。
この現象を無くす方法は、第２部割込み編　で説明します。
この章では、総合デモ（メロディ）の簡単な説明をしたいと思います。

それでは、プログラムリストを見てみましょう！

２．プログラムリスト

　このサンプルは、前章に１モジュール追加して、８ファイルの構成になっています。

“ｖｅｃｔｏｒＡ．ａｓｍ”	前章のまま使用したので解説を省略します。
“Ｓｔａｒｔｕｐ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＰｉｏ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＬｃｄ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＴｉｍｅ．ｃ”	前章のまま使用したので解説を省略します。
“ＣａｔＳｕｂ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＵｓａｒｔ．ｃ”	前章のまま使用したので解説を省略します。
“ＰｏｌＭｅｌｏｄｙ．ｃ”	メロディのコントロール部です。
“Ｃａｔ２０４ｐ．ｃ”	メインコントロール部です。

1）メロディのコントロール部

file “PolMelody.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  ポーリング                                                 */
  4: /*                                                                           */
  5: /*  ＜MOD＞       PolMelody.c                                                */
  6: /*  ＜役割＞      デモ メロディー関係                                        */
  7: /*  ＜TAB＞       ４タブ編集                                                 */
  8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                             */
  9: /*  ＜使用ﾊｰﾄﾞ＞  CAT-261-H8S/2612 エーワン(株)                              */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <string.h>
 13: #include    "io261x.h"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*          音階 Hz                                                          */
 17: /*****************************************************************************/
 18: #define dO          262             /* _ド */
 19: #define rE          293             /* _レ  */
 20: #define mI          330             /* _ミ  */
 21: #define fhA         349             /* _ﾌｧ */
 22: #define sO          392             /* _ソ  */
 23: #define rA          440             /* _ﾗ  */
 24: #define sI          494             /* _シ  */
 25: #define do          523             /* ド  */
 26: #define re          587             /* レ   */
 27: #define mi          659             /* ミ   */
 28: #define fha         698             /* ﾌｧ  */
 29: #define so          784             /* ソ   */
 30: #define ra          880             /* ﾗ   */
 31: #define si          987             /* シ   */
 32: #define Do          1047            /* ド  */
 33:
```

```
34: #define SCMAX       32                  /* 楽譜テーブルの最大数              */
35: /*****************************************************************************/
36: /*      変数宣言                                                             */
37: /*****************************************************************************/
38:         Uchar       MelStep;            /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                       */
39:         Uchar       MelMode;            /* 演奏ﾓｰﾄﾞ                          */
40:         Uchar       MelSelect;          /* 自動選曲                          */
41:         Ushort      MelodyHz;           /* Melody Hz                         */
42:         Uchar       Music;              /* 自動演奏ｶｳﾝﾀｰ                     */
43:         Ushort      Doremi[SCMAX];      /* ﾄﾞﾚﾐ音階周波数(Hz)の RAM 側        */
44:         Ushort      Rhythm[SCMAX];      /* ﾘｽﾞﾑ(msec)                         */
45: const   Ushort      DoremiTbl[] =       /* ﾄﾞﾚﾐ音階周波数(Hz)                 */
46: {
47:         do,         /* ド */
48:         re,         /* レ  */
49:         mi,         /* ミ  */
50:         fha,        /* ﾌｧ */
51:         so,         /* ソ  */
52:         ra,         /* ﾗ  */
53:         si,         /* シ  */
54:         Do,         /* ド */
55:         0
56: };
57: const   Ushort      MusicTbl[3][SCMAX] =    /* 自動演奏の楽譜 (最大 32 ﾏﾃﾞ)   */
58: {                                                    /*  ﾈｺﾌﾝｼﾞｬｯﾀ   */
59:     { ra, so, do, Do, Do, ra, so, do, Do, Do,
60:       ra, so, do, Do, rA, Do, sO, si, si},
61:                                                      /*  ｲﾇﾉｵﾏﾜﾘｻﾝ   */
62:     { mi, do, do, do, mi, do, do, do,fha,fha, mi, mi, re,
63:      fha,fha, mi, mi, re, re, ra, ra, so,fha, mi, re, do},
64:                                                      /*  ｱﾏﾘﾘｽ       */
65:     { so, ra, so, Do, so, ra, so, ra, ra, so, ra, so,fha, mi, re, mi, do, 0},
66: };
67: const   Ushort      RhythmTbl[3][SCMAX] =   /* 自動演奏のリズム(最大 32 ﾏﾃﾞ) */
68: {                                                    /*  ﾈｺﾌﾝｼﾞｬｯﾀ   */
69:     {250,250,500,500,500,250,250,500,500,500,
```

```
 70:       250,250,500,500,500,500,500,500,500},
 71:                                                                    /*  ｲﾇﾉｵﾏﾜﾘｻﾝ    */
 72:      {250,250,250,250,250,250,250,250,250,250,250,250,250,1000,
 73:       250,250,250,250,250,250,250,250,250,250,250,250,250,1000,},
 74:                                                                    /*  ｱﾏﾘﾘｽ        */
 75:      {500,500,500,500,500,500,1000,500,500,500,500,250,250,250,250,500,500,  0},
 76: };
 77: const   Uchar       *MelodyTblA[] =     /*  自動演奏の曲名                    */
 78: {
 79:         ″          ″,
 80:         ″ﾈｺﾌﾝｼﾞｬｯﾀ″,
 81:         ″ｲﾇﾉｵﾏﾜﾘｻﾝ″,
 82:         ″ｱﾏﾘﾘｽ     ″,
 83: };
 84: /*************************************************************************/
 85: /*       Ｍｅｍ初期化                                                    */
 86: /*************************************************************************/
 87: void    MelMemInitial(void)
 88: {
 89:     MelStep   = 0;                  /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                    */
 90:     MelMode   = 0;                  /* 演奏ﾓｰﾄﾞ                        */
 91:     MelSelect = 0;                  /* 自動選曲                        */
 92:     MelodyHz  = 0;                  /* Melody Min Hz                   */
 93: }
 94: /*************************************************************************/
 95: /*       Ｉ／Ｏ初期化                                                    */
 96: /*************************************************************************/
 97: void    MelIoInitial(void)
 98: {
 99:     Buzzer(0);                      /* Buzer OFF                       */
100: }
101: /*************************************************************************/
102: /*       Melody          ﾒﾛﾃﾞｨｰｺﾝﾄﾛｰﾙ                                  */
103: /*************************************************************************/
104: void    Melody()
105: {
```

```
106:     MelodySequence();                  /* ﾒﾛﾃﾞｨｰ操作ｺﾝﾄﾛｰﾙ                */
107:     if (MelMode == 0) ManualMelody();  /* 手動演奏                        */
108:     else              AutoMelody();    /* 自動演奏                        */
109: }
110: /*****************************************************************************/
111: /*      MelodySequence()    ﾒﾛﾃﾞｨｰ操作ｺﾝﾄﾛｰﾙ                                */
112: /*****************************************************************************/
113: void    MelodySequence()
114: {
115:     Uchar   port;
116:
117:     port = GetUpPort(1);               /* PB[PA0]->PB[PF7]                */
118:     if (port & 0xe) {                  /* PB[PC2]->PB[PF7] ON(立上) ?      */
119:         Buzzer(0);                     /* Buzer OFF                       */
120:         if (port & 0x2) {              /* PB[PC2] ON ? ﾓｰﾄﾞ変更           */
121:             MelStep = 0;               /* 強制停止                        */
122:             if (MelMode == 0) {        /* 現在手動 ?                      */
123:                 MelMode   = 1;         /* 自動に変更                      */
124:                 MelSelect = 1;         /* 自動選曲                        */
125:             }
126:             else {                     /* 現在自動 ?                      */
127:                 MelMode   = 0;         /* 手動に変更                      */
128:                 MelSelect = 0;         /* 自動選曲                        */
129:             }
130:         }
131:         if (MelMode != 0) {            /* 自動 ?                          */
132:             if (port & 0x4) {          /* PB[PC5] ON ?  選曲              */
133:                 MelStep = 0;           /* 強制停止                        */
134:                 if (++MelSelect > 3) MelSelect = 1;
135:             }
136:             if (port & 0x8) {          /* PB[PF7] ON ? 自動演奏開始       */
137:                 if (MelStep == 0) MelStep = 1;      /*  開始             */
138:                 else              MelStep = 0;      /*  停止             */
139:                 Music   = 0;
140:             }
141:         }
```

```
142:         if (MelMode == 0) GotoxyMemSet(4,1,"M");     /* 手動表示              */
143:         else              GotoxyMemSet(4,1,"A");     /* 自動表示              */
144:         GotoxyMemSet(7,0,(Uchar *)MelodyTblA[MelSelect]);   /* 曲名表示      */
145:     }
146: }
147: /*****************************************************************************/
148: /*      ManualMelody   手動ﾒﾛﾃﾞｨｰ演奏                                       */
149: /*****************************************************************************/
150: void    ManualMelody()
151: {
152:     Uchar   port;
153:
154:     switch(MelStep) {
155:     case 0:
156:         _strcpyW(Doremi,(Ushort *)DoremiTbl);   /* ﾄﾞﾚﾐ音階周波数(初期準備) */
157:         MelodyHz  = 0;                  /* Melody Min Hz                     */
158:         MelStep++;
159:         break;
160:     case 1:
161:         if (GetInPort(0) & 0xff) {      /* P40->P47 どれかがＯＮしたか？     */
162:             port = GetUpPort(0);
163:             if (port & 0x1) {           /* SW[P40] 立上がり ON (ド)          */
164:                 Buzzer(MelodyHz = Doremi[7]);
165:             }
166:             else if (port & 0x2) {      /* SW[P41] 立上がり ON (レ)          */
167:                 Buzzer(MelodyHz = Doremi[6]);
168:             }
169:             else if (port & 0x4) {      /* SW[P42] 立上がり ON (ミ)          */
170:                 Buzzer(MelodyHz = Doremi[5]);
171:             }
172:             else if (port & 0x8) {      /* SW[P43] 立上がり ON (ファ)        */
173:                 Buzzer(MelodyHz = Doremi[4]);
174:             }
175:             else if (port & 0x10) {     /* SW[P44] 立上がり ON (ソ)          */
176:                 Buzzer(MelodyHz = Doremi[3]);
177:             }
```

```
178:             else if (port & 0x20) {     /* SW[P45] 立上がり ON (ラ)          */
179:                 Buzzer(MelodyHz = Doremi[2]);
180:             }
181:             else if (port & 0x40) {     /* SW[P46] 立上がり ON (シ)          */
182:                 Buzzer(MelodyHz = Doremi[1]);
183:             }
184:             else if (port & 0x80) {     /* SW[P47] 立上がり ON (ド)          */
185:                 Buzzer(MelodyHz = Doremi[0]);
186:             }
187:         }
188:         else if (MelodyHz != 0) {       /* Buzzer 停止                       */
189:             Buzzer(MelodyHz = 0);
190:         }
191:         break;
192:     }
193: }
194: /*************************************************************************/
195: /*      AutoMelody    自動ﾒﾛﾃﾞｨｰ演奏                                     */
196: /*************************************************************************/
197: void    AutoMelody()
198: {
199:     switch(MelStep) {
200:     case 0:
201:         break;
202:     case 1:
203:         _strcpyW(Doremi,(Ushort *)&MusicTbl[MelSelect-1][0]);
204:         _strcpyW(Rhythm,(Ushort *)&RhythmTbl[MelSelect-1][0]);
205:         ++MelStep;
206:         break;
207:     case 2:
208:         MelodyHz = Doremi[Music];       /*  楽譜        */
209:         if (MelodyHz != 0) {            /*  演奏中      */
210:             Buzzer(MelodyHz);
211:             ++MelStep;
212:         }
213:         else {
```

```
214:             Buzzer(0);                  /*  停止        */
215:             Music = 0;
216:             MelStep = 4;
217:         }
218:         break;
219:     case 3:                              /*  リズム      */
220:         SoftWait1ms(Rhythm[Music]);
221:         Music++;
222:         MelStep = 2;
223:         break;
224:     case 4:
225:         SoftWait1ms(500);               /*  連続時の間  */
226:         MelStep = 2;
227:         break;
228:     }
229: }
```

［リストの説明］

１８～３２行：

音階ごとの周波数をシンボル定義しました。

３４行：

自動演奏での音符数の最大数宣言です。

３８～４４行：

このモジュールで使用する変数の宣言です。

４５～５６行：

マニュアル演奏用ドレミ…の音階周波数テーブルです。

５７～６６行：

自動演奏３曲の音階周波数テーブルです。

６７～７６行：

自動演奏３曲のリズム（音の長さ）ｍｓ時間テーブルです。

７７～８３行：

自動演奏曲名のテーブルです。

８７～９３行：

このモジュールで使用する変数の初期化関数です。

メインのメモリ初期化の時に呼ばれます。

９７～１００行：

このモジュールで使用するＩ／Ｏの初期化関数です。

メインのＩ／Ｏ初期化の時に呼ばれます。

１０４～１０９行：

手動／自動演奏を切り換えをコントロールする関数です。

１１３～１４６行：

メロディを動作させる場合、ＰＢ操作を管理する関数です。

１５０～１９３行：

手動演奏を制御する関数です。

１９７～２２９行：

自動演奏を制御する関数です。

2）メインコントロール

file “Cat261p.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  ポーリング                                                 */
  4: /*                                                                           */
  5: /*  ＜MOD＞       Cat261p.c                                                  */
  6: /*  ＜役割＞      main                                                       */
  7: /*  ＜TAB＞       ４タブ編集                                                 */
  8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                             */
  9: /*  ＜使用ﾊｰﾄﾞ＞ CAT-261-H8S/2612 エーワン(株)                               */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    "io261x.h"
 13: #include    "DemoCtl.h"
 14: /*****************************************************************************/
 15: /*      外部変数使用宣言                                                     */
 16: /*****************************************************************************/
 17: extern  short       BuzzerHz;          /* Buzzer Hz                        */
 18: extern  Uchar       SciStep;           /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
 19: extern  Uchar       SciSelect;         /* ﾎﾞｰﾚｰﾄ選択                       */
 20: extern  Uchar       MelStep;           /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
 21: extern  Uchar       MelMode;           /* 演奏ﾓｰﾄﾞ                         */
 22: extern  Uchar       MelSelect;         /* 自動選曲                         */
 23: /*****************************************************************************/
 24: /*      変数宣言                                                             */
 25: /*****************************************************************************/
 26:         Uchar       ModeStep;          /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
 27:         Uchar       Shift;             /* shift ﾊﾟﾀｰﾝ                       */
 28: /*****************************************************************************/
 29: /*      _main()                                                              */
 30: /*****************************************************************************/
 31: void    _main(void)  {
 32:     disable();                         /* PBC の割込の為                   */
 33:     SYSCR = 0x21;                      /* ｼｽﾃﾑｺﾝﾄﾛｰﾗ 割込ﾓｰﾄﾞ 2           */
```

```
34:                                     /*              NMI 立下りエッジ        */
35:                                     /*              RAME 有効               */
36:     SoftWait1ms(400);               /* 400msWait(ﾘｾｯﾄ遅延時間ﾊｰﾄﾞ)          */
37:                                     /* H-ﾃﾞﾊﾞｯｶﾞ<->Target 通信可能に成るまで*/
38:                                     /* の 1 回リトライ時間分待つ(20 回)     */
39:     MemInitial();                   /* ﾒﾓﾘ系初期化                          */
40:     IoInitial();                    /* I/O 系初期化                        */
41:     while(1) {
42:         SigInput();                 /* Signal Input  Process               */
43:         SoftWait1ms(20);            /*         ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り       */
44:         ModeCntrol();               /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                          */
45:         AllLcdDisp();               /* LCD     全画面表示                  */
46:         SigOutput();                /* Signal Output Process(LED 点灯)     */
47:     }
48: }
49: /*****************************************************************************/
50: /*      Ｍｅｍ初期化                                                         */
51: /*****************************************************************************/
52: void    MemInitial(void)
53: {
54:     ModeStep = 0;                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                    */
55:     Shift = 0;                      /* Led Disp Patan Initial              */
56:
57:     PioMemInitial();                /* PIO     Mem 初期化                  */
58:     LcdMemInitial();                /* LCD     Mem 初期化                  */
59:     TimMemInitial();                /* TIM     Mem 初期化                  */
60:     SciMemInitial();                /* SCI     Mem 初期化                  */
61:     MelMemInitial();                /* Melody Mem 初期化                   */
62: }
63: /*****************************************************************************/
64: /*      Ｉ／Ｏ初期化                                                         */
65: /*****************************************************************************/
66: void    IoInitial(void)
67: {
68:     PioIoInitial();                 /* PIO     I/O 初期化                  */
69:     LcdIoInitial();                 /* LCD     I/O 初期化                  */
```

```
70:     TimIoInitial();                 /* TIM    I/O 初期化              */
71:     SciIoInitial();                 /* SCI    I/O 初期化              */
72:     MelIoInitial();                 /* Melody I/O 初期化              */
73: }
74: /*************************************************************************/
75: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                       */
76: /*************************************************************************/
77: void    ModeCntrol()
78: {
79:     if (GetUpPort(1) & 0x1) {       /* PB[PA0] ON?(立上)               */
80:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST    */
81:         else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST    */
82:         else if (ModeStep < 30) ModeStep = 30;      /*  SCI   Goto TEST    */
83:         else if (ModeStep < 40) ModeStep = 40;      /*  Demo  Melody       */
84:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ     */
85:         Buzzer(0);                                  /*  強制 OFF           */
86:     }
87:     switch(ModeStep) {
88:     case 0:
89:         GotoxyMemSet(0,0,"CAT261&AH6000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ      */
90:         GotoxyMemSet(0,1,"Polling    [PA0]");
91:         ModeStep++;
92:         break;
93:     case 1:
94:         RunRun();                   /* ｼﾌﾄLED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ      */
95:         break;
96:     case 10:                        /*  PIO   TEST                     */
97:         GotoxyMemSet(0,0,"PIO         ");
98:         GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
99:         ModeStep++;
100:        break;
101:    case 11:
102:        PioDemo();
103:        break;
104:    case 20:                        /*  Timer TEST                     */
105:        GotoxyMemSet(0,0,"Timer/Counter   ");
```

```
106:        GotoxyMemSet(0,1,"0000Hz[+PF7-PC5]");
107:        BuzzerHz = 0;                   /* Buzzer Hz                    */
108:        ModeStep++;
109:        break;
110:    case 21:
111:        TimerDemo();
112:        RunRun();                       /* ｼﾌﾄLED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ      */
113:        break;
114:    case 30:                            /* SCI TEST                     */
115:        GotoxyMemSet(0,0,"SCI    8,n,1[PC2]");
116:        GotoxyMemSet(0,1,"09600b[+PF7-PC5]");
117:        SciStep   = 0;                  /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                  */
118:        SciSelect = 1;                  /* 9600BPS                      */
119:        ModeStep++;
120:        break;
121:    case 31:
122:        SciDemo();
123:        RunRun();                       /* ｼﾌﾄLED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ      */
124:        break;
125:    case 40:                            /*  Demo  Melody                */
126:        GotoxyMemSet(0,0,"Melody          ");
127:        GotoxyMemSet(0,1,"PC2[M]F7[ｽ]C5[ｾ]");
128:        MelStep   = 0;                  /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                  */
129:        MelMode   = 0;                  /* 演奏ﾓｰﾄﾞ                     */
130:        MelSelect = 0;                  /* 自動選曲                     */
131:        ModeStep++;
132:        break;
133:    case 41:
134:        Melody();
135:        RunRun();
136:        break;
137:    }
138: }
139: /*****************************************************************************/
140: /*      RunRun()        CPU 走行表示                                         */
141: /*****************************************************************************/
```

```
142: void    RunRun()
143: {
144:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
145:     PutOutPort(Shift,'=');
146: }
147: /*****************************************************************************/
148: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                     */
149: /*****************************************************************************/
150: void    SoftWait1ms(Ushort ms)
151: {
152:     while(ms-- != 0) {
153:         Wait1ms();
154:     }
155: }
156: /*****************************************************************************/
157: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                   */
158: /*****************************************************************************/
159: void    SoftWait10us(Ushort us)
160: {
161:     while(us-- != 0) {
162:         Wait10us();
163:     }
164: }
165: /*****************************************************************************/
166: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                   */
167: /*****************************************************************************/
168: void    Wait1ms()
169: {
170:     asm("       push.w  r0                  ");
171:     asm("       mov.w   #5000,r0            ");      /* 5000*4=20000cyc  */
172:     asm(" wait:                             ");
173:     asm("       dec.w   #1,r0               ");      /* 1 clock          */
174:     asm("       bne     wait:16             ");      /* 3 clock          */
175:     asm("       pop.w   r0                  ");
176: }
177: /*****************************************************************************/
```

```
178: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                */
179: /*****************************************************************************/
180: void    Wait10us()
181: {
182:     asm("       push.w  r0                      ");
183:     asm("       mov.w   #50,r0                  ");      /* 50*4=200cyc      */
184:     asm(" wait1:                                ");
185:     asm("       dec.w   #1,r0                   ");      /* 1 clock          */
186:     asm("       bne     wait1:16                ");      /* 3 clock          */
187:     asm("       pop.w   r0                      ");
188: }
```

［リストの説明］　　前章のメインコントロールから追加された部分だけ解説します。

２０～２２行：

このモジュールで使用する外部変数宣言を追加しました。

６１行：

“ＰｏｌＭｅｌｏｄｙ．ｃ”で使用する変数の初期化関数を呼んでいます。

７２行：

“ＰｏｌＭｅｌｏｄｙ．ｃ”で使用するＩ／Ｏを初期化する関数を呼んでいます。

１２５～１３６行：

この章のサンプルプログラムを動作させるための制御部分を追加しました。

これで、この章のリスト説明は終わりです。

自動演奏曲“ネコフンジャッタ”を聞いて気がついた方もいらしゃるかと思いますが、
半音の登録をしていないため、チョット編曲をしてしまいました。
興味の有る方は、半音登録をして作りなおしてみて下さい。
これも、プログラムに慣れ親しむ方法として、よいことかもしれません。

この章の前半でも説明しましたが、全てポーリングで作成しているため動作および反応が鈍い部分があります。
このままで市場に出したらクレームになってしまいます。
この動作を改善するために、第２部　割込み編より修正を進めていきたいと思います。

第２部　割込み編
第１章　タイマ割込み

割り込みモード２は、Ｈ－ｄｅｂａｇｇｅｒがＨ８Ｓ／２６１２の**ハードトレース機能**を使用するにあたり必要なモードです。(他のモードでは、**ハードトレース**が機能しません)
それとＣＰＵ内部機能の**ＰＢＣ**（ＰＣブレークコントローラ）とを組み合わせてデバッグ可能にしています。(割り込みモード０の場合は、**ソフトトレース**とＰＢＣの組み合わせになります。)
最初の章からＣＰＵに対して割り込みモード２の設定と割り込み要求マスクレベル（６以下）の設定は済んでいます。（ＰＢＣ割り込みのプライオリティが７の為）
この章では、割り込みモード２を使用したアプリケーションをどのような手続きで追加するかをリストに沿って説明を進めていきます。
割り込み要因として、ＴＰＵ＿０での連続コンペアマッチにおける１０ｍｓ毎割り込み例を使用しました。なお、改造するサンプルの元は、“第１部ポーリング編－第７章総合デモ”で作成したサンプルを使用します。

動作仕様の変更はしませんので、この部より動作説明は省略しますが、
￥評価ボード￥第２部割込み編￥第１章タイマ割込み￥
にファイラーでディレクトリの移動をしておいて下さい。

＜補足説明＞

１）**ハードトレース**とは、ＣＰＵ機能が持っているトレース割り込みを利用した機能です。ただしこの機能を利用する場合は、割り込みモードを「２」にする必要があります。

２）**ソフトトレース**とは、ソフトにより次の実行命令を判定した後、ＰＢＣに次命令のアドレスを設定したトレース機能になります。このソフトトレースはＰＢＣ搭載したＣＰＵで、割り込みモード０の場合とハードトレース機能が無い場合に使用します。

１．ベクターテーブルとスタートアップを割り込み用に変更する。

Ｈ８Ｓシリーズは、すべての割り込み要因に対し独立したベクタアドレスが割り当てられていますので、要因別ベクタテーブルへの登録が必要になります。
又，モジュール別のインタラプトプライオリティレジスタも持っていますのでこの設定も必要になります。
リストの説明に入る前にインタラプトプライオリティレジスタ関連と、割り込み要求マスクレベルとの関係について説明します。

１）割り込みモード２の設定

システムコントロールレジスタ（ＳＹＳＣＲ）				
ﾋﾞｯﾄ	名前	初期値	R/W	機能
7	MACS	1	R/W	ＭＡＣサチュレーション
6	－	0	－	リザーブビット
5 4	INTM1 INTM0	0 0	R/W R/W	００：割り込み制御モード０ ０１：禁止 **１０：割り込み制御モード２** １１：禁止
3	NMIEG	0	R/W	０：ＮＭＩ立下りで受け付け １：ＮＭＩ立上りで受け付け
2	－	0	－	リザーブビット
1	－	0	－	リザーブビット
0	RAME	1	R/W	０：内蔵ＲＡＭ無効 １：内蔵ＲＡＭ有効

表［２・１－１－１］

２）割り込み要求マスクレベルの設定

ＣＰＵ内部のエクステンドレジスタ（ＥＸＲ）に設定				
ﾋﾞｯﾄ	名前	初期値	R/W	機能
7	T	0	R/W	トレースビット
6~3	－	1	-	リザーブビット
2 1 0	I2 I1 I0	1 1 1	R/W R/W R/W	**割り込み要求マスクレベル（０～７）を指定します。**

表［２・１－１－２］

3）インタラプトプライオリティの設定

レジスタ名	ビット			
	7	6～4	3	2～0
ＩＰＲＡ	0	ＩＲＱ０　（0～7）の設定	0	ＩＲＱ１　（0～7）の設定
ＩＰＲＢ	0	ＩＲＱ２ ＩＲＱ３　（0～7）の設定	0	ＩＲＱ４ ＩＲＱ５　（0～7）の設定
ＩＰＲＣ	0		0	ＤＴＣ　（0～7）の設定
ＩＰＲＤ	0	ＷＯＶＩ０（0～7）の設定	0	
ＩＰＲＥ	0	**ＰＢＣ　（7）　の設定 （H-debugger で使用する）**	0	ＡＤＩ　（0～7）の設定
ＩＰＲＦ	0	ＴＰＵ＿０（0～7）の設定	0	ＴＰＵ＿１（0～7）の設定
ＩＰＲＧ	0	ＴＰＵ＿２（0～7）の設定	0	ＴＰＵ＿３（0～7）の設定
ＩＰＲＨ	0	ＴＰＵ＿４（0～7）の設定	0	ＴＰＵ＿５（0～7）の設定
ＩＰＲＪ	0		0	ＳＣＩ＿０（0～7）の設定
ＩＰＲＫ	0	ＳＣＩ＿１（0～7）の設定	0	ＳＣＩ＿２（0～7）の設定
ＩＰＲＭ	0	ＨＣＡＮ　（0～7）の設定	0	ＭＭＴ　（0～7）の設定

表［２・１－１－３］　ＩＰＲｘ

(1)ＥＸＲレジスタ”６”の状態で、ＰＢＣ以外の割り込みを不許可にしたい場合は、**プライオリティを”６”**もしくは各モジュールで割り込みをマスク状態にして下さい。

(2)ＥＸＲレジスタ”0”にしますと、全割り込み許可状態になります。

4）**割り込み制御モード２の割り込み受け付けシーケンス**

割り込み発生（要因　インタラプトプライオリティ）

```
{
    ｉｆ（割り込み　＝＝　ＮＭＩ）                    割り込み受け付け（）；
    ｅｌｓｅ　ｉｆ（インタラプトプライオリティ　＝＝　７）　｛
            ｉｆ　（割り込み要求マスクレベル　＜＝　６）　割り込み受け付け（）；
            ｅｌｓｅ                                  保留（）；
    }
    ｅｌｓｅ　ｉｆ（インタラプトプライオリティ　＝＝　６）　｛
            ｉｆ　（割り込み要求マスクレベル　＜＝　５）　割り込み受け付け（）；
            ｅｌｓｅ                                  保留（）；
    }
    ＜一部省略＞
    ｅｌｓｅ　ｉｆ（インタラプトプライオリティ　＝＝　１）　｛
            ｉｆ　（割り込み要求マスクレベル　＜＝　０）　割り込み受け付け（）；
            ｅｌｓｅ                                  保留（）；
    }
}
```

１）ベクターテーブルの変更箇所

file “vectorA.asm”

```
  1: ;*************************************************************************
  2: ;*
  3: ;*  Ｈ８Ｓ  デバッカモニター用ベクターテーブル(GNU/as)
  4: ;*              TYPE:  H8S/2132.2134.2138.2612.2633F-ZTAT
  5: ;*              MODE:  Advanced
  6: ;*                                      2002/03/14 by AONE  M.Hasegawa
  7: ;*************************************************************************
  8:             .h8300s
  9:             .file       "vectorA.asm"
 10:             .extern     _StartUp
 11:             .extern     _non
 12:             .extern     _UserTGIA0
 14:             .section    .vector
<<<  省略  >>>
 22: ;*******************************************************
 23: ;*  ベクターテーブル
 24: ;*******************************************************
 25: _VectorTable:                       ;           (共通/213x)   (26xx)
 26:             VECTOR      _StartUp    ;* VCT(  0) RESET       |
 27:             VECTOR      _non        ;* VCT(  1) System 予約 1|
<<<  省略  >>>
 56:             VECTOR      _non        ;* VCT( 30) ﾘｻﾞｰﾌﾞ       |
 57:             VECTOR      _non        ;* VCT( 31) ﾘｻﾞｰﾌﾞ       |
 58:             VECTOR      _UserTGIA0  ;* VCT( 32) ﾘｻﾞｰﾌﾞ       |TPU_0 TGIA_0
 59:             VECTOR      _non        ;* VCT( 33) ﾘｻﾞｰﾌﾞ       |      TGIB_0
<<<  省略  >>>
153:             VECTOR      _non        ;* VCT(127) ﾘｻﾞｰﾌﾞ       |      TEI2
154:             .end
```

［リストの説明］

５８行：

ＴＰＵ__０のＴＧＩＡ__０コンペアマッチ割り込み要因処理のＣ言語関数“Ｕｓｅｒ ＴＧＩＡ０ ()”のアドレスを登録しました。

２）スタートアップの変更

file “Startup.c”

```
  1: /************************************************************************/
  2: /* ＜サンプル＞     割り込み                                            */
  4: /*       ＜MOD＞     Startup.c                                          */
  5: /*       ＜タブ＞     ４タブ編集       GNU/gcc                          */
  6: /*                    スタートアップ（CAT261-H8S/2612)                  */
  7: /************************************************************************/
  8: /************************************************************************/
  9: /*       StartUp                                                        */
 10: /************************************************************************/
 11: void    StartUp()
 12: {
 13:     asm("mov.l  #0xffefbe,sp");          /* ｽﾀｯｸﾎﾟｲﾝﾀ 設定              */
 14:     asm("jmp    _main       ");          /* void _main()                */
 15: }
＜＜＜  省略  ＞＞＞
 25: /************************************************************************/
 26: /*       UserTGIA0                                                      */
 27: /************************************************************************/
 28: #pragma interrupt
 29: void    UserTGIA0()
 30: {
 31:     TimerA0();
 32: }
```

［リストの説明］

２８～３２行：

ＴＧＩＡ_0コンペアマッチ割り込みハンドラ関数になります。

割り込み例外処理を示す“＃ｐｒａｇｍａ　ｉｎｔｅｒｒｕｐｔ”を定義することのより、関数の入口／出口に全レジスターの退避／復帰命令が挿入され、かつリターン命令が“ＲＴＥ”になる便利な定義です。

Ｃ言語用記述での割り込みハンドラ雛型になります。

２．タイマモジュールを割り込み用に変更する。

ＴＰＵ_0での連続コンペアマッチにおける１０ｍｓ毎割り込みを使用する場合、ＴＰＵ_0の初

期設定が必要になります。
また、割り込みハンドラから呼ばれる関数“ＴｉｍｅｒＡ０ ()”も追加しました。
この関数は、タイマー割り込みを利用した内部ソフトタイマー機能を処理する役割になります。

リストの説明に入る前に、ＴＰＵ＿０関係資料を添付します。

タイマスタートレジスタ（ＴＳＴＲ）					
ﾋﾞｯﾄ	名前	初期値	R/W	機能	
7 6	－	0 0	－	リザーブ	
5 4 3 2 1 0	CST5 CST4 CST3 CST2 CST1 CST0	0 0 0 0 0 0	R/W R/W R/W R/W R/W R/W	ＴＰＵ＿５ ＴＰＵ＿４ ＴＰＵ＿３ ＴＰＵ＿２ ＴＰＵ＿１ ＴＰＵ＿０	０：カウンターをストップします １：カウンターをスタートします

表［２・１－２－１］　ＴＳＴＲ

タイマコントロールレジスタ（ＴＣＲ＿０）				
ﾋﾞｯﾄ	名前	初期値	R/W	機能
7 6 5	CCLR2 CCLR1 CCLR0	0 0 0	R/W R/W R/W	０００：ＴＣＮＴ＿０のクリア禁止 **００１：ＴＧＲＡ＿０のコンペアマッチでＴＣＮＴ＿０クリア** ０１０：ＴＧＲＢ＿０のコンペアマッチでＴＣＮＴ＿０クリア ０１１：同期クリア／同期動作をしている他のクリアでクリア １００：ＴＣＮＴ＿０のクリア禁止 １０１：ＴＧＲＣ＿０のコンペアマッチでＴＣＮＴ＿０クリア １１０：ＴＧＲＤ＿０のコンペアマッチでＴＣＮＴ＿０クリア １１１：同期クリア／同期動作をしている他のクリアでクリア
4 3	CKEG1 CKEG0	0 0	R/W R/W	**００：立上りエッジでカウント** ０１：立下りエッジでカウント １ｘ：両エッジでカウント
2 1 0	TPSC2 TPSC1 TPSC0	0 0 0	R/W R/W R/W	ＴＣＮＴ＿０のカウンタクロックソースの選択 ０００：内部　φクロック **００１：　　φ／４クロック** ０１０：　　φ／１６クロック ０１１：　　φ／６４クロック １００：外部　ＴＣＬＫＡ端子入力でカウント １０１：　　ＴＣＬＫＢ端子入力でカウント １１０：　　ＴＣＬＫＣ端子入力でカウント １１１：　　ＴＣＬＫＤ端子入力でカウント

表［２・１－２－２］　ＴＣＲ＿０

タイマインタラプトイネーブルレジスタ（ＴＩＥＲ＿０）				
ﾋﾞｯﾄ	名前	初期値	R/W	機能

7	TTGE	0	R/W	A／D変換開始要求イネーブル ０：A／D変換開始要求禁止 １：A／D変換開始要求許可
6	－	1	－	リザーブビット
5	TCIEU	0	R/W	ＴＰＵ＿１，２，４，５のアンダーフローイネーブル ０：ＴＣＦＵによる要求禁止 １：ＴＣＦＵによる要求許可
4	TGIEV	0	R/W	オーバーフローイネーブル ０：ＴＣＦＶによる要求禁止 １：ＴＣＦＶによる要求許可
3	TGIED	0	R/W	ＴＰＵ＿０，３のＴＧＲＤ割込み要求イネーブル ０：ＴＧＦＤによる要求禁止 １：ＴＧＦＤによる要求許可
2	TGIEC	0	R/W	ＴＰＵ＿０，３のＴＧＲＣ割込み要求イネーブル ０：ＴＧＦＣによる要求禁止 １：ＴＧＦＣによる要求許可
1	TGIEB	0	R/W	ＴＧＲＢ割込み要求イネーブル ０：ＴＧＦＢによる要求禁止 １：ＴＧＦＢによる要求許可
0	TGIEA	0	R/W	ＴＧＲＡ割込み要求イネーブル ０：ＴＧＦＡによる要求禁止 **１：ＴＧＦＡによる要求許可**

表［２・１－２－３］　ＴＩＥＲ＿０

タイマステータスレジスタ（ＴＳＲ＿０）				
ﾋﾞｯﾄ	名前	初期値	R/W	機能
7	TCFD	1	R	ＴＣＮＴのカウント方向フラグ ０：ダウンカウント　１：アップカウント
6	－	1	－	リザーブビット
5	TCFU	0	R/W	１（R）：アンダーフローによる要求フラグです。 ０（W）：要求フラグ解除になります。
4	TCFV	0	R/W	１（R）：オーバーフローによる要求フラグです。 ０（W）：ＴＣＦＶによる要求解除になります。
3	TGFD	0	R/W	１（R）：ＴＰＵ＿０，３のＴＧＲＤによる要求フラグです。 ０（W）：ＴＧＦＤによる要求解除になります。
2	TGFC	0	R/W	１（R）：ＴＰＵ＿０，３のＴＧＲＣによる要求フラグです。 ０（W）：ＴＧＦＣによる要求解除になります。
1	TGFB	0	R/W	１（R）：ＴＧＲＢによる要求フラグです。 ０（W）：ＴＧＦＢによる要求解除になります。
0	TGFA	0	R/W	１（R）：ＴＧＲＡによる要求フラグです。 **０（W）：ＴＧＦＡによる要求解除になります。**

表［２・１－２－４］　ＴＳＲ＿０

１）プログラムリスト

file “IntrTime.c”

```
/*****************************************************************************/
/*  ＜サンプル＞  割込み                                                     */
/*                                                                           */
/*  ＜MOD＞       IntrTime.c                                                 */
/*  ＜役割＞      タイマ関係                                                 */
/*  ＜TAB＞       ４タブ編集                                                 */
/*  ＜保守ﾂｰﾙ＞   makefile 参照                                              */
/*  ＜使用ﾊｰﾄﾞ＞  CAT-261-H8S/2612 エーワン(株)                               */
/*****************************************************************************/
#include    <string.h>
#include    "io261x.h"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      ブザー（タイマ）関係のマクロ                                         */
/*****************************************************************************/
#define BZMIN       200                 /* Buzzer   Min 200Hz               */
#define BZMAX       1100                /*          Max 1100Hz              */
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        short       BuzzerHz;           /* Buzzer Hz                        */
        Uchar       TmUp[TMMAX];        /* Soft Timer Up ﾌﾗｸﾞ                */
        Uchar       TmSt[TMMAX];        /*            Start ﾌﾗｸﾞ             */
        Ushort      TmCnt[TMMAX];       /*            Count                 */
/*****************************************************************************/
/*      Ｍｅｍ初期化                                                         */
/*****************************************************************************/
void    TimMemInitial(void)
{
    BuzzerHz = 0;                       /* Buzzer Hz                        */
    memset(TmSt, OFF,sizeof(TmSt));     /* Timer(10ms) Initial              */
    memset(TmUp, OFF,sizeof(TmUp));
    memset(TmCnt,OFF,sizeof(TmCnt));
}
/*****************************************************************************/
```

```
/*        Ｉ／Ｏ初期化                                                          */
/*************************************************************************/
void    TimIoInitial(void)
{
    MSTPCRA &= ~0x20;                   /* ﾓｼﾞｭｰﾙｽﾄｯﾌﾟ  TPU ｽﾀｰﾄ              */
    IPRF    = 0x57;                     /* TPU_0 ﾌﾟﾗｲｵﾘﾃｨ 5                    */
    TCR_0   = 0x21;                     /*       TGRA ｺﾝﾍﾟｱﾏｯﾁで TCNT ｸﾘｱ  001 */
                                        /*       立上りｴｯｼﾞ                 00  */
                                        /*       sys/4(5,000,000Hz)      001 */
    TIER_0  = 1;                        /*       TGIEA=1 TGRA により割込み  */
    TCNT_0  = 0;                        /*       TCNT=0                     */
    TGRA_0  = 50000;                    /*       (20000000/4)/50000=100(Hz) */
    TSTR    |= 1;                       /*       TCNT0 ｶｳﾝﾄ動作              */
}
/*************************************************************************/
/*      Buzzer   TPU_3  TIOCA3 出力に Buzzer 接続  sys = 20,000,000Hz        */
/*************************************************************************/
void    Buzzer(Ushort hz)
{
    Ushort   cyc;

    if ((hz >= BZMIN) && (hz <= BZMAX)) {   /* Buzzer ON                  */
        TCR_3 = 0x21;                   /* CNTL TGRA ｺﾝﾍﾟｱﾏｯﾁで TCNT ｸﾘｱ  001 */
                                        /*       立上りｴｯｼﾞ                 00  */
                                        /*       sys/4(5,000,000Hz)      001 */
        TMDR_3 = 0xc2;                  /* MODE Default                  1100 */
                                        /*       PWM ﾓｰﾄﾞ 1                0010 */
        TIORH_3 = 3;                    /* I/O  TGRB 未使用              0000 */
                                        /*       TIOCA3 ﾏｯﾁﾄｸﾞﾙ出力        0011 */
        cyc = 5000000 / hz;             /* 周期の計算                         */
        TGRA_3 = cyc / 2;               /* 周期設定/2                         */
        TCNT_3 = 0;                     /* ｶｳﾝﾄｸﾘｱ                            */
        TSTR  |= 0x8;                   /* TCNT_3 ｽﾀｰﾄ                        */
    }
    else {                              /* Buzzer OFF                 */
        TCNT_3 = 0;                     /* ｶｳﾝﾄｸﾘｱ                            */
```

```
74:         TSTR  &= ~0x8;                  /* TCNT_3 ｽﾄｯﾌﾟ                    */
75:   }
76: }
77: /*************************************************************************/
78: /*      TmStart                                                          */
79: /*************************************************************************/
80: void    TmStart(short tno,short ms)
81: {
82:     Ushort  cnt;
83:
84:     cnt = ms / 10;                  /* 単位 10ms に修正する             */
85:     TmUp[tno]  = OFF;
86:     TmCnt[tno] = cnt;
87:     TmSt[tno]  = ON;
88: }
89: /*************************************************************************/
90: /*      TmUpTest    ﾀｲﾏ UP ﾌﾗｸﾞﾃｽﾄ                                       */
91: /*************************************************************************/
92: Uchar    TmUpTest(short tno)
93: {
94:     return(TmUp[tno]);
95: }
96: /*************************************************************************/
97: /*      ＴＭＡ０　タイマー割り込み（10ms）　割り込みハンドラより呼ばれる    */
98: /*************************************************************************/
99: void    TimerA0(void)
100: {
101:     short   i;
103:     TSR_0 &= ~(1);                 /*  TGFA=0(ｺﾝﾍﾟｱﾌﾗｸﾞ Reset)          */
105:     for(i = 0;i < TMMAX;i++) {     /*  内部タイマー処理                 */
106:         if (TmSt[i] == ON) {
107:             if (--TmCnt[i] == 0) {
108:                 TmUp[i] = ON;
109:                 TmSt[i] = OFF;
110:             }
111:         }
```

```
112:     }
113: }
114: /*****************************************************************************/
115: /*      TimerDemo   Timer デモ                                               */
116: /*****************************************************************************/
117: void    TimerDemo()
118: {
119:     Uchar   port;
120:     Uchar   dec[4+1];
121:
122:     port = GetUpPort(1);                /* PB[PA0]->PB[PF7]                 */
123:     if (port & 0xc) {                   /* PB[PC5] | PB[PF7] ON ?           */
124:         if (port & 0x4) {               /* PB[-PC5] ON-立上り                */
125:             BuzzerHz -= 100;
126:         }
127:         else if (port & 0x8) {          /* PB[+PF7] ON-立上り                */
128:             BuzzerHz += 100;
129:         }
130:         if (BuzzerHz < BZMIN) BuzzerHz = BZMIN;
131:         if (BuzzerHz > BZMAX) BuzzerHz = BZMAX;
132:         Buzzer(BuzzerHz);
133:         Bin2AdecN(dec,BuzzerHz,4);      /* 表示用データ作成                 */
134:         GotoxyMemSet(0,1,dec);
135:     }
136: }
```

［リストの説明］　前章から変更のあった部分をおもに解説します。

２４～２６行：

タイマー割り込みを利用した内部ソフトタイマー処理を追加しましたので、その処理に使用するフラグおよびカウンタの役目をする変数を宣言する。

３０～３６行：

このモジュールで使用する変数の初期化関数です。

メインのメモリ初期化の時に呼ばれます。

４０～５１行：

タイマ／カウンタを初期化する関数です。

メインのＩ／Ｏ初期化の時に呼ばれます。

ここでは、ＴＰＵ＿０とＩＰＲＦの初期化設定をします。

［４２行］

モジュールストップコントロールレジスタのＴＰＵをオフ（スタート）しています。

［４３行］

ＴＰＵ＿０のインタラプトプライオリティを“５”に設定しています。

別に意味はありません“６”以下でしたら構いません。

表［２・１－１－３］参照

［４４行］

タイマコントロールレジスタの初期設定をしています。

１）ＴＧＲＡコンペアマッチでＴＣＮＴ＿０をクリアする。

２）立上りエッジでカウントする。

３）システムクロック（２０ＭＨｚ）の１／４でカウントする

表［２・１－２－２］参照

［４７行］

タイマインタラプトイネーブルレジスタの設定をしています。

１）ＴＧＲＡのＴＧＦＡビットによる割り込み要求の許可をセット

表［２・１－２－３］参照

［４８行］

カウンタＴＣＮＴ＿０をゼロにしています。（必要ありませんが念のため）

［４９行］

１００Ｈｚ（１０ｍｓ）に１回ＴＧＦＡが立つようにコンペアデータをＴＧＲＡ＿０に設定しています。

設定値（５００００）＝（（２０ＭＨｚ／４）／１００Ｈｚ）

［５０行］

ＴＣＮＴ＿０のカウント動作指示をタイマスタートレジスタにセットしています。

表［２・１－２－１］参照

８０～８８行：

タイマー割り込みを利用した内部ソフトタイマーのスタート関数です。

９２～９５行：

タイマー割り込みを利用した内部ソフトタイマーのタイムアップしたかを調べる関数です。

９９～１１３行：

タイマー割り込みを利用した内部ソフトタイマー処理関数です。

ＴＰＵ＿０のＴＧＩＡ＿０コンペアマッチ割り込み

タイマＢチャンル２のタイマー割り込み処理で“Startup. c”の割り込みハンドラーから呼ばれています。

［１０３行］

ＴＧＲＡのＴＧＦＡビットによる割り込み要求解除の意味でタイマーステータスレジスタのＴＧＦＡビットをクリアしています。

連続割り込みが必要な場合は、必ずクリアして下さい。

表［２・１－２－４］参照

３．メインコントロール部を割り込み用に変更する。

割り込みを可能にするために必要な変更および、タイマー割り込みを利用した内部ソフトタイマーを利用するために、割り込み許可の挿入とメインループ処理の変更をしています

プログラムに沿って説明します。

1）プログラムリスト

file “Cat261i.c”

```
/*****************************************************************************/
/*                                                                           */
/*  ＜サンプル＞ 割込み                                                      */
/*                                                                           */
/*  ＜MOD＞      Cat261i.c                                                    */
/*  ＜役割＞     main                                                         */
/*  ＜TAB＞      ４タブ編集                                                   */
/*  ＜保守ﾂｰﾙ＞  makefile 参照                                                */
/*  ＜使用ﾊｰﾄﾞ＞ CAT-261-H8S/2612 エーワン(株)                                */
/*                                                                           */
/*****************************************************************************/
#include    "io261x.h"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      外部変数使用宣言                                                     */
/*****************************************************************************/
extern  short       BuzzerHz;           /* Buzzer Hz                        */
extern  Uchar       SciStep;            /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
extern  Uchar       SciSelect;          /* ﾎﾞｰﾚｰﾄ選択                       */
extern  Uchar       MelStep;            /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
extern  Uchar       MelMode;            /* 演奏ﾓｰﾄﾞ                         */
extern  Uchar       MelSelect;          /* 自動選曲                         */
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       ModeStep;           /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
```

```
27:         Uchar       Shift;              /* shift ﾊﾟﾀｰﾝ                          */
28: /*****************************************************************************/
29: /*      _main()                                                              */
30: /*****************************************************************************/
31: void    _main(void)  {
32:     disable();                          /* PBC の割込の為                      */
33:     SYSCR = 0x21;                       /* ｼｽﾃﾑｺﾝﾄﾛｰﾗ 割込ﾓｰﾄﾞ 2              */
34:                                         /*             NMI 立下りエッジ        */
35:                                         /*             RAME 有効               */
36:     SoftWait1ms(400);                   /* 400msWait(ﾘｾｯﾄ遅延時間ﾊｰﾄﾞ)          */
37:                                         /* H-ﾃﾞﾊﾞｯｶﾞ<->Target 通信可能に成るまで*/
38:                                         /* の 1 回リトライ時間分待つ(20 回)     */
39:     MemInitial();                       /* ﾒﾓﾘ系初期化                         */
40:     IoInitial();                        /* I/O 系初期化                        */
41:     TmStart(TM0,20);                    /* ﾁｬﾀﾘﾝｸﾞ防止ｽﾀｰﾄ                     */
42:     enable();
43:     while(1) {
44:         if (TmUpTest(TM0) == ON) {
45:             TmStart(TM0,20);            /* ﾁｬﾀﾘﾝｸﾞ防止ｽﾀｰﾄ                     */
46:             SigInput();                 /* Signal Input  Process               */
47:             ModeCntrol();               /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                          */
48:             SigOutput();                /* Signal Output Process               */
49:         }
50:         AllLcdDisp();                   /* LCD    全画面表示                   */
51:     }
52: }
53: /*****************************************************************************/
54: /*      Ｍｅｍ初期化                                                         */
55: /*****************************************************************************/
56: void    MemInitial(void)
57: {
58:     ModeStep = 0;                       /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                    */
59:     Shift = 0;                          /* Led Disp Patan Initial              */
60:
61:     PioMemInitial();                    /* PIO    Mem 初期化                   */
62:     LcdMemInitial();                    /* LCD    Mem 初期化                   */
```

```
63:     TimMemInitial();                    /* TIM    Mem 初期化              */
64:     SciMemInitial();                    /* SCI    Mem 初期化              */
65:     MelMemInitial();                    /* Melody Mem 初期化              */
66: }
67: /*****************************************************************************/
68: /*       Ｉ／Ｏ初期化                                                        */
69: /*****************************************************************************/
70: void    IoInitial(void)
71: {
72:     PioIoInitial();                     /* PIO    I/O 初期化              */
73:     LcdIoInitial();                     /* LCD    I/O 初期化              */
74:     TimIoInitial();                     /* TIM    I/O 初期化              */
75:     SciIoInitial();                     /* SCI    I/O 初期化              */
76:     MelIoInitial();                     /* Melody I/O 初期化              */
77: }
78: /*****************************************************************************/
79: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                           */
80: /*****************************************************************************/
81: void    ModeCntrol()
82: {
83:     if (GetUpPort(1) & 0x1) {           /* PB[PA0] ON?(立上)              */
84:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST    */
85:         else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST    */
86:         else if (ModeStep < 30) ModeStep = 30;      /*  SCI   Goto TEST    */
87:         else if (ModeStep < 40) ModeStep = 40;      /*  Demo  Melody       */
88:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ     */
89:         Buzzer(0);                                  /*  強制 OFF           */
90:     }
91:     switch(ModeStep) {
92:     case 0:
93:         GotoxyMemSet(0,0,"CAT261&AH6000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ      */
94:         GotoxyMemSet(0,1,"Interrupt  [PA0]");
95:         ModeStep++;
96:         break;
97:     case 1:
98:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ      */
```

```
 99:         break;
100:     case 10:                           /*  PIO   TEST                    */
101:         GotoxyMemSet(0,0,"PIO             ");
102:         GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
103:         ModeStep++;
104:         break;
105:     case 11:
106:         PioDemo();
107:         break;
108:     case 20:                           /*  Timer TEST                    */
109:         GotoxyMemSet(0,0,"Timer/Counter   ");
110:         GotoxyMemSet(0,1,"0000Hz[+PF7-PC5]");
111:         BuzzerHz = 0;                 /* Buzzer Hz                      */
112:         ModeStep++;
113:         break;
114:     case 21:
115:         TimerDemo();
116:         RunRun();                     /* ｼﾌﾄLED 点灯 OUTﾊﾞｯﾌｧｰにｾｯﾄ       */
117:         break;
118:     case 30:                           /* SCI TEST                       */
119:         GotoxyMemSet(0,0,"SCI   8,n,1[PC2]");
120:         GotoxyMemSet(0,1,"09600b[+PF7-PC5]");
121:         SciStep   = 0;                /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                    */
122:         SciSelect = 1;                /* 9600BPS                        */
123:         ModeStep++;
124:         break;
125:     case 31:
126:         SciDemo();
127:         RunRun();                     /* ｼﾌﾄLED 点灯 OUTﾊﾞｯﾌｧｰにｾｯﾄ       */
128:         break;
129:     case 40:                           /*  Demo  Melody                  */
130:         GotoxyMemSet(0,0,"Melody          ");
131:         GotoxyMemSet(0,1,"PC2[M]F7[ｽ]C5[ｾ]");
132:         MelStep   = 0;                /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                    */
133:         MelMode   = 0;                /* 演奏ﾓｰﾄﾞ                       */
134:         MelSelect = 0;                /* 自動選曲                       */
```

```
135:         ModeStep++;
136:         break;
137:     case 41:
138:         Melody();
139:         RunRun();
140:         break;
141:     }
142: }
143: /*************************************************************************/
144: /*      RunRun()        CPU 走行表示                                     */
145: /*************************************************************************/
146: void    RunRun()
147: {
148:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
149:     PutOutPort(Shift,'=');
150: }
151: /*************************************************************************/
152: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                 */
153: /*************************************************************************/
154: void    SoftWait1ms(Ushort ms)
155: {
156:     while(ms-- != 0) {
157:         Wait1ms();
158:     }
159: }
160: /*************************************************************************/
161: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                               */
162: /*************************************************************************/
163: void    SoftWait10us(Ushort us)
164: {
165:     while(us-- != 0) {
166:         Wait10us();
167:     }
168: }
169: /*************************************************************************/
170: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait               */
```

```
/************************************************************************/
void    Wait1ms()
{
    asm("       push.w  r0                      ");
    asm("       mov.w   #5000,r0                ");      /* 5000*4=20000cyc  */
    asm(" wait:                                 ");
    asm("       dec.w   #1,r0                   ");      /* 1 clock          */
    asm("       bne     wait:16                 ");      /* 3 clock          */
    asm("       pop.w   r0                      ");
}
/************************************************************************/
/*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (20.000MHz)  Non Wait                 */
/************************************************************************/
void    Wait10us()
{
    asm("       push.w  r0                      ");
    asm("       mov.w   #50,r0                  ");      /* 50*4=200cyc      */
    asm(" wait1:                                ");
    asm("       dec.w   #1,r0                   ");      /* 1 clock          */
    asm("       bne     wait1:16                ");      /* 3 clock          */
    asm("       pop.w   r0                      ");
}
```

［リストの説明］ 前章から変更のあった部分をおもに解説します。

４１行：

チャタリング防止タイマーに、タイマー割り込みを利用した内部ソフトタイマーの利用に変更しますので、ここでスタートをかけておきます。

４２行：

ここで、割り込み許可“ｅｎａｂｌｅ ()”をします。

割り込み要求マスクレベルをゼロ（０）にしています。 “ｉｏ２６１ｘ．ｈ”を参照

逆を言えば、ここに来るまでに割り込み関数で使用するＩ／Ｏおよび変数は初期化ずみであることが必要になります。

４４～４５行：

チャタリング防止タイマがタイムアップするのを待ちます。

いままで使用していたソフトタイマと違い、待っている間は他処理の実行が可能になっています。

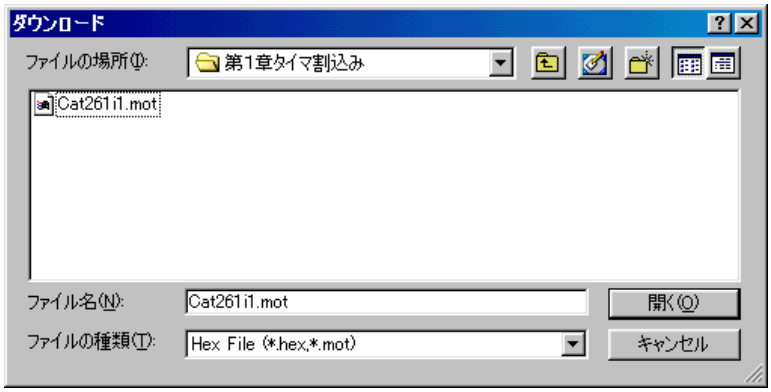
４６～最終行：

変更なし。

以上、４ファイルの変更で、タイマー割り込み処理が組み込まれました。

動作的には、ほとんど変わりませんが興味のあるかたは、動作確認してみて下さい。

ＤＥＦでダウンロード後、プログラム実行をして下さい。



“Ｃａｔ２６１ｉ１．ｍｏｔ”です

次章は、ＳＣＩ割り込みに挑戦します。

第2章　ＳＣＩ割込み

　ここの章では、ＳＣＩの送受信を割り込みで処理する場合どのような手続きが必要かをリストに沿って説明を進めていきます。
また、送受信フォームを文字列通信に変更しましたので、文字列扱いを容易にするため簡単なプロトコルでの通信仕様にしました。

［プロトコル仕様］

ＳＴＸ “文字列～１４文字まで” ＥＴＸ　ＳＴＸ＝０ｘ０２（スタート）
ＥＴＸ＝０ｘ０３（エンド）

ＳＣＩ０（文字列送信）―＞ＳＣＩ１（文字列受信）
ＳＣＩ１（文字列送信）―＞ＳＣＩ０（文字列受信）　の方式でループバック通信を繰り返します。

￥評価ボード￥第２部割込み編￥第２章ＳＣＩ割込み￥
に、ファイラーでディレクトリ移動をしておいて下さい。

１．ベクターテーブルとスタートアップをＳＣＩ割り込み用に変更する。

１）ベクターテーブルの変更箇所

file “vectorA.asm”

```
   1: ;*********************************************************************************
<<<  省略  >>>
  26: ;*****************************************************
  27: ;*  ベクターテーブル
  28: ;*****************************************************
  29: _VectorTable:                       ;          (共通/213x)    (26xx)
<<<  省略  >>>
 109:            VECTOR      _non         ;* VCT( 79) ﾘｻﾞｰﾌﾞ       |
 110:            VECTOR      _UserRXI0    ;* VCT( 80) SCI0  ERI0   |
 111:            VECTOR      _UserRXI0    ;* VCT( 81)       RXI0   |
 112:            VECTOR      _UserTXI0    ;* VCT( 82)       TXI0   |
 113:            VECTOR      _non         ;* VCT( 83)       TEI0   |
 114:            VECTOR      _UserRXI1    ;* VCT( 84) SCI1  ERI1   |
 115:            VECTOR      _UserRXI1    ;* VCT( 85)       RXI1   |
 116:            VECTOR      _UserTXI1    ;* VCT( 86)       TXI1   |
 117:            VECTOR      _non         ;* VCT( 87)       TEI1   |
<<<  省略  >>>
 154:            .end
```

［リストの説明］

１１０～１１１行：

ＳＣＩ０のＥＲＩ０（受信エラー）とＲＸＩ０（受信完了）の割り込み要因処理時のＣ言語関数“ＵｓｅｒＲＸＩ０ ()”のアドレスを登録しました。

１１２行：

ＳＣＩ０のＴＸＩ０（送信データエンプティ）の割り込み要因処理時のＣ言語関数“ＵｓｅｒＴＸＩ０ ()”のアドレスを登録しました。

１１４～１１５行：

ＳＣＩ１のＥＲＩ１（受信エラー）とＲＸＩ１（受信完了）の割り込み要因処理時のＣ言語関数“ＵｓｅｒＲＸＩ１ ()”のアドレスを登録しました。

１１６行：

ＳＣＩ１のＴＸＩ１（送信データエンプティ）の割り込み要因処理時のＣ言語関数“ＵｓｅｒＴＸＩ１ ()”のアドレスを登録しました。

2）スタートアップの変更

file “Startup.c”

```
/************************************************************************/
/* ＜サンプル＞     割り込み                                            */
/*                                                                      */
/*      ＜MOD＞     Startup.c                                           */
/*      ＜タブ＞     ４タブ編集       GNU/gcc                           */
/*                  スタートアップ (CAT261-H8S/2612)                    */
/************************************************************************/
/************************************************************************/
/*      StartUp                                                         */
/************************************************************************/
void    StartUp()
{
    asm("mov.l  #0xffefbe,sp");         /* ｽﾀｯｸﾎﾟｲﾝﾀ 設定              */
    asm("jmp    __main      ");         /* void _main()                 */
}
```

＜＜＜　省略　＞＞＞

```
/************************************************************************/
/*      UserRXI0                                                        */
/************************************************************************/
#pragma interrupt
void    UserRXI0()
{
    Rxi0();
}
/************************************************************************/
/*      UserTXI0                                                        */
/************************************************************************/
#pragma interrupt
void    UserTXI0()
{
    Txi0();
}
/************************************************************************/
```

```
50: /*      UserRXI1                                                        */
51: /************************************************************************/
52: #pragma interrupt
53: void    UserRXI1()
54: {
55:     Rxi1();
56: }
57: /************************************************************************/
58: /*      UserTXI1                                                        */
59: /************************************************************************/
60: #pragma interrupt
61: void    UserTXI1()
62: {
63:     Txi1();
64: }
```

［リストの説明］

３６～４０行：

ＳＣＩ０のＥＲＩ０（受信エラー）とＲＸＩ０（受信完了）の割り込みハンドラ関数です。

４４～４８行：

ＳＣＩ０のＴＸＩ０（送信データエンプティ）の割り込みハンドラ関数です。

５２～５６行：

ＳＣＩ１のＥＲＩ１（受信エラー）とＲＸＩ１（受信完了）の割り込みハンドラ関数です。

６０～６４行：

ＳＣＩ１のＴＸＩ１（送信データエンプティ）の割り込みハンドラ関数です。

２．ＳＣＩモジュールを割り込み用に変更する。

割り込みに対応するため、割り込み用送受信関数が必要になります。
また、ポーリング時は１文字での送受信ループバック動作でしたが、今回は割り込みサンプルですので、文字列での送受信ループバック連続動作をさせてみます。

プログラムに沿って説明します。

１）プログラムリスト

file “IntrSci.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  割込み                                                     */
  4: /*                                                                           */
  5: /*  ＜MOD＞       IntrSci.c                                                  */
  6: /*  ＜役割＞      SCI(SIO) 関係                                              */
  7: /*  ＜TAB＞       ４タブ編集                                                 */
  8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                             */
  9: /*  ＜使用ﾊｰﾄﾞ＞  CAT-204-KC5C8012 エーワン(株)                              */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <string.h>
 13: #include    "io261x.h"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*      SCI ボレート計算+etc                                                 */
 17: /*                  Sys   = 20000000Hz                                   */
 18: /*                  32    = 64*2(0-1)                                    */
 19: /*****************************************************************************/
 20: #define CLOCK       (20000000/32)       /* 調歩同期 n=0 時の計算式        */
 21: #define BRRN(b)     ((CLOCK/b)-1)
 22: #define NON         0                   /* ﾊﾟﾘﾃｨ    NON                  */
 23: #define EVEN        1                   /*           EVEN                 */
 24: #define ODD         2                   /*           ODD                  */
 25: #define STX         0x2                 /* 送信スタートコード             */
```

```
26: #define ETX         0x3                 /* 送信ストップコード                 */
27: /*****************************************************************************/
28: /*      変数宣言                                                             */
29: /*****************************************************************************/
30:         Uchar       SciStep;            /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                        */
31:         Uchar       SciSelect;          /* ﾎﾞｰﾚｰﾄ選択                          */
32:         Uchar       TxDat0[16];         /* SCI0 送信ﾃﾞｰﾀ                      */
33:         Uchar       TxIdx0;             /*      送信中 ｲﾝﾃﾞｯｸｽ                 */
34:         Uchar       TxCnt0;             /*      送信残 ｶｳﾝﾀ                   */
35:         Uchar       RxDat0[16];         /*      受信ﾃﾞｰﾀ                      */
36:         Uchar       RxIdx0;             /*      受信中 ｲﾝﾃﾞｯｸｽ                 */
37:         Uchar       RxEnd0;             /*      受信終了ﾌﾗｸﾞ                   */
38:         Uchar       TxDat1[16];         /* SCI1 送信ﾃﾞｰﾀ                      */
39:         Uchar       TxIdx1;             /*      送信中 ｲﾝﾃﾞｯｸｽ                 */
40:         Uchar       TxCnt1;             /*      送信残 ｶｳﾝﾀ                   */
41:         Uchar       RxDat1[16];         /*      受信ﾃﾞｰﾀ                      */
42:         Uchar       RxIdx1;             /*      受信中 ｲﾝﾃﾞｯｸｽ                 */
43:         Uchar       RxEnd1;             /*      受信終了ﾌﾗｸﾞ                   */
44: const   Ushort      BpsTbl[] =
45: {
46:         4800,
47:         9600,
48:         19200,
49:         31250,
50:         38400,
51:         0,
52: };
53: /*****************************************************************************/
54: /*      Ｍｅｍ初期化                                                         */
55: /*****************************************************************************/
56: void    SciMemInitial(void)
57: {
58:     SciStep   = 0;                      /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                        */
59:     SciSelect = 1;                      /* 9600BPS                            */
60: }
61: /*****************************************************************************/
```

```
/*       Ｉ／Ｏ初期化                                                    */
/*************************************************************************/
void    SciIoInitial(void)
{
    RsOpen0(9600,8,NON);                /* SCI0 Open                    */
    RsOpen1(9600,8,NON);                /* SCI1 Open                    */
}
/*************************************************************************/
/*  RS232C(SCI0) Open bps    = ﾎﾞﾚｰﾄ[4800,*9600,19200,31250,38400]      */
/*                    ch     = ｷｬﾗｸﾀ[7,*8]                              */
/*                    pty    = ﾊﾟﾘﾃｨ[*0=NON 1=EVEN 2=ODD]               */
/*                    *      = ﾃﾞﾌｫﾙﾄ                                   */
/*                    固定   = x1 STOP=1bit                             */
/*************************************************************************/
void    RsOpen0(Ushort bps,Uchar ch,Uchar pty)
{
    Uchar   bck;
    Uchar   smr;

    MSTPCRB &= ~0x80;                   /* ﾓｼﾞｭｰﾙｽﾄｯﾌﾟ SCI0 ｽﾀｰﾄ        */
    IPRJ    = 0x76;                     /* SCI0 ﾌﾟﾗｲｵﾘﾃｨ 6               */
    SCMR_0 = 0xf2;                      /* ｽﾏｰﾄ SMIF=0 Non ｽﾏｰﾄ          */

    smr = 0;                            /* ﾓｰﾄﾞ ｷｬﾗｸﾀ[8bit] ﾊﾟﾘﾃｨ[NON]  */
    if (ch == 7)          smr |= 0x40;  /*      ｷｬﾗｸﾀ[7bit]             */
    if (pty == EVEN)      smr |= 0x20;  /*      ﾊﾟﾘﾃｨ[EVEN]             */
    else if (pty == ODD) smr |= 0x30;   /*      ﾊﾟﾘﾃｨ[ODD]              */
    SMR_0 = smr;

    if (bps == 4800)        bck = BRRN(4800);
    else if (bps == 9600)   bck = BRRN(9600);
    else if (bps == 19200)  bck = BRRN(19200);
    else if (bps == 31250)  bck = BRRN(31250);
    else if (bps == 38400)  bck = BRRN(38400);
    BRR_0 = bck;                        /* ﾎﾞｰﾚｰﾄｼﾞｪﾈﾚｰﾄ設定            */
    Wait1ms();                          /* 1bit 経過待ち 2400BPS(0.42ms+a)  */
```

```
 98:     SSR_0 &= 0x87;                          /* RDRF+ResetStatus                 */
 99:     SCR_0 = 0x70;                           /* RIE(RX ｲﾝﾀﾗﾌﾟﾄｲﾈｰﾌﾞﾙ)+TE+RE      */
100: }
101: /*****************************************************************************/
102: /*  RS232C(SCI1) Open bps    = ﾎﾞﾚｰﾄ[4800,*9600,19200,31250,38400]           */
103: /*                     ch     = ｷｬﾗｸﾀ[7,*8]                                   */
104: /*                     pty    = ﾊﾟﾘﾃｨ[*0=NON 1=EVEN 2=ODD]                    */
105: /*                     *      = ﾃﾞﾌｫﾙﾄ                                        */
106: /*                     固定   = x1 STOP=1bit                                  */
107: /*****************************************************************************/
108: void    RsOpen1(Ushort bps,Uchar ch,Uchar pty)
109: {
110:     Uchar   bck;
111:     Uchar   smr;
112:
113:     MSTPCRB &= ~0x40;                       /* ﾓｼﾞｭｰﾙｽﾄｯﾌﾟ SCI1 ｽﾀｰﾄ            */
114:     IPRK    = 0x67;                         /* SCI1 ﾌﾟﾗｲｵﾘﾃｨ 6                  */
115:     SCMR_1 = 0xf2;                          /* ｽﾏｰﾄ SMIF=0 Non ｽﾏｰﾄ             */
116:
117:     smr = 0;                                /* ﾓｰﾄﾞ ｷｬﾗｸﾀ[8bit] ﾊﾟﾘﾃｨ[NON]      */
118:     if (ch == 7)          smr |= 0x40;      /*      ｷｬﾗｸﾀ[7bit]                 */
119:     if (pty == EVEN)      smr |= 0x20;      /*      ﾊﾟﾘﾃｨ[EVEN]                 */
120:     else if (pty == ODD) smr |= 0x30;       /*      ﾊﾟﾘﾃｨ[ODD]                  */
121:     SMR_1 = smr;
122:
123:     if (bps == 4800)         bck = BRRN(4800);
124:     else if (bps == 9600)    bck = BRRN(9600);
125:     else if (bps == 19200)   bck = BRRN(19200);
126:     else if (bps == 31250)   bck = BRRN(31250);
127:     else if (bps == 38400)   bck = BRRN(38400);
128:     BRR_1 = bck;                            /* ﾎﾞｰﾚｰﾄｼﾞｪﾈﾚｰﾄ設定               */
129:     Wait1ms();                              /* 1bit 経過待ち 2400BPS(0.42ms+a)  */
130:     SSR_1 &= 0x87;                          /* RDRF+ResetStatus                 */
131:     SCR_1 = 0x70;                           /* RIE(RX ｲﾝﾀﾗﾌﾟﾄｲﾈｰﾌﾞﾙ)+TE+RE      */
132: }
133: /*****************************************************************************/
```

```
/*       TxStart0     SCI0 文字列送信(TXI0 割込み準備)                    */
/*************************************************************************/
void   TxStart0(Uchar *tx,Uchar cnt)
{
    TxDat0[0] = STX;                    /* スタートコード                  */
    memcpy(&TxDat0[1],tx,cnt);          /* 送信ﾊﾞｯﾌｧｰに記憶                */
    TxDat0[cnt+1] = ETX;                /* ストップコード                  */
    TxIdx0 = 1;                         /* 送信中ｲﾝﾃﾞｯｸｽ                   */
    TxCnt0 = cnt+1;                     /* 送信残ｶｳﾝﾀ                     */

    TDR_0 = TxDat0[0];                  /* 送信ﾃﾞｰﾀｾｯﾄ                     */
    SSR_0 &= ~(0x80);                   /* TDRE=0                          */
    SCR_0 |= 0x80;                      /* TIE=1 TX ｲﾝﾀﾗﾌﾟﾄｲﾈｰﾌﾞﾙ           */
}
/*************************************************************************/
/*       TxStart1     SCI1 文字列送信(TXI1 割込み準備)                    */
/*************************************************************************/
void   TxStart1(Uchar *tx,Uchar cnt)
{
    TxDat1[0] = STX;                    /* スタートコード                  */
    memcpy(&TxDat1[1],tx,cnt);          /* 送信ﾊﾞｯﾌｧｰに記憶                */
    TxDat1[cnt+1] = ETX;                /* ストップコード                  */
    TxIdx1 = 1;                         /* 送信中ｲﾝﾃﾞｯｸｽ                   */
    TxCnt1 = cnt+1;                     /* 送信残ｶｳﾝﾀ                     */

    TDR_1 = TxDat1[0];                  /* 送信ﾃﾞｰﾀｾｯﾄ                     */
    SSR_1 &= ~(0x80);                   /* TDRE=0                          */
    SCR_1 |= 0x80;                      /* TIE=1 TX ｲﾝﾀﾗﾌﾟﾄｲﾈｰﾌﾞﾙ           */
}
/*************************************************************************/
/*       Txi0         TXI0 割込み       割り込みハンドラより呼ばれる      */
/*************************************************************************/
void    Txi0()
{
    if (TxCnt0 == 0) SCR_0 &= ~0x80;    /* TIE=0 TX ｲﾝﾀﾗﾌﾟﾄﾃﾞｨｾｰﾌﾞﾙ         */
    else {
```

```
170:        TDR_0 = TxDat0[TxIdx0];         /* 送信ﾃﾞｰﾀｾｯﾄ                    */
171:        SSR_0 &= ~(0x80);               /* TDRE=0 TSR へ転送               */
172:        ++TxIdx0;
173:        --TxCnt0;
174:    }
175: }
176: /*************************************************************************/
177: /*      Txi1        TXI1 割込み      割り込みハンドラより呼ばれる          */
178: /*************************************************************************/
179: void    Txi1()
180: {
181:    if (TxCnt1 == 0) SCR_1 &= ~0x80;    /* TIE=0 TX ｲﾝﾀﾗﾌﾟﾄﾃﾞｨｾｰﾌﾞﾙ          */
182:    else {
183:        TDR_1 = TxDat1[TxIdx1];         /* 送信ﾃﾞｰﾀｾｯﾄ                    */
184:        SSR_1 &= ~(0x80);               /* TDRE=0 TSR へ転送               */
185:        ++TxIdx1;
186:        --TxCnt1;
187:    }
188: }
189: /*************************************************************************/
190: /*      Rxi0        RXI0 割込み        割り込みハンドラより呼ばれる          */
191: /*************************************************************************/
192: void    Rxi0()
193: {
194:    Uchar  dt;
195:
196:    dt = RDR_0;                         /* 受信                            */
197:    if (SSR_0 & 0x38) {                 /* SCI  OE+FE+PE Error             */
198:        SSR_0 &= 0x87;                  /* RDRF+Error Reset                */
199:        RxIdx0 = 0;
200:    }
201:    else {                              /* 正常受信                        */
202:        SSR_0 &= ~(0x40);               /* RDRF=0                          */
203:        if (dt == STX)      RxIdx0 = 0;     /* スタートコード              */
204:        else if (dt == ETX) RxEnd0 = ON;    /* 受信終了ﾌﾗｸﾞｾｯﾄ             */
205:        else {                          /* 受信データ                      */
```

```
206:             RxDat0[RxIdx0++] = dt;
207:             if (RxIdx0 >= sizeof(RxDat0)) RxIdx0 = 0;   /* ﾊﾞﾌｧｵｰﾊﾞｰの防止 */
208:         }
209:     }
210: }
211: /*************************************************************************/
212: /*      Rxi1        RXI0 割込み        割り込みハンドラより呼ばれる          */
213: /*************************************************************************/
214: void    Rxi1()
215: {
216:     Uchar  dt;
217:
218:     dt = RDR_1;                         /* 受信                          */
219:     if (SSR_1 & 0x38) {                 /* SCI  OE+FE+PE Error           */
220:         SSR_1 &= 0x87;                  /* RDRF+Error Reset              */
221:         RxIdx1 = 0;
222:     }
223:     else {                              /* 正常受信                      */
224:         SSR_1 &= ~(0x40);               /* RDRF=0                        */
225:         if (dt == STX)      RxIdx1 = 0;     /* スタートコード            */
226:         else if (dt == ETX) RxEnd1 = ON;    /* 受信終了ﾌﾗｸﾞｾｯﾄ           */
227:         else {                          /* 受信データ                    */
228:             RxDat1[RxIdx1++] = dt;
229:             if (RxIdx1 >= sizeof(RxDat1)) RxIdx1 = 0;   /* ﾊﾞﾌｧｵｰﾊﾞｰの防止 */
230:         }
231:     }
232: }
233: /*************************************************************************/
234: /*      SciDemo()      ＵＳＡＲＴデモ                                    */
235: /*************************************************************************/
236: void    SciDemo()
237: {
238:     Uchar   dec[2+1];
239:     short   stat;
240:
241:     SciSequence();                      /* SCI 操作ｺﾝﾄﾛｰﾙ                */
```

```
242:    switch(SciStep) {
243:    case 0:
244:        break;
245:    case 1:
246:        RsOpen0(BpsTbl[SciSelect],8,NON);   /* RS232C(SCI0) Open          */
247:        RsOpen1(BpsTbl[SciSelect],8,NON);   /* RS232C(SCI1) Open          */
248:        GotoxyMemSet(0,0,"xxxxxxxxxxx");
249:        SciStep++;
250:        break;
251:    case 2:
252:        RxEnd1 = OFF;                   /* SCI1 受信終了ﾌﾗｸﾞ              */
253:        TmStart(TM1,500);               /*      ﾀｲﾑｵｰﾊﾞｰ管理               */
254:        TxStart0(" n e r p S ",11);     /* SCI0 送信/受信割込み開始       */
255:        SciStep++;
256:        break;
257:    case 3:
258:        if (RxEnd1 == ON) {             /* SCI1 受信終了を待つ            */
259:            RxDat1[RxIdx1] = 0;         /*      確認用の表示データ作成    */
260:            GotoxyMemSet(0,0,RxDat1);
261:            SciStep++;
262:        }
263:        else if (TmUpTest(TM1) == ON) { /* SCI1 ﾀｲﾑｵｰﾊﾞｰ                  */
264:            GotoxyMemSet(0,0,"ﾀｲﾑｵｰﾊﾞｰS0 ");    /* Error 表示             */
265:            SciStep++;
266:        }
267:        break;
268:    case 4:
269:        RxEnd0 = OFF;                   /* SCI0 受信終了ﾌﾗｸﾞ              */
270:        TmStart(TM1,500);               /*      ﾀｲﾑｵｰﾊﾞｰ管理               */
271:        TxStart1("I t r u t C",11);     /* SCI1 送信/受信割込み開始       */
272:        SciStep++;
273:        break;
274:    case 5:
275:        if (RxEnd0 == ON) {             /* SCI0 受信終了を待つ            */
276:            RxDat0[RxIdx0] = 0;         /*      確認用の表示データ作成    */
277:            GotoxyMemSet(0,0,RxDat0);
```

```
278:            SciStep = 2;
279:        }
280:        else if (TmUpTest(TM1) == ON) { /* SCI1 ﾀｲﾑｵｰﾊﾞｰ                    */
281:            GotoxyMemSet(0,0,"ﾀｲﾑｵｰﾊﾞｰS1 ");    /* Error 表示                 */
282:            SciStep = 2;
283:        }
284:        break;
285:    }
286: }
287: /*****************************************************************************/
288: /*      SciSequence()   Usart 操作ｺﾝﾄﾛｰﾙ                                      */
289: /*****************************************************************************/
290: void    SciSequence()
291: {
292:     Uchar   port;
293:     Uchar   dec[5+1];
294:
295:     port = GetUpPort(1);                /* PB[PA0]->PB[PF7]                 */
296:     if (port & 0xe) {                   /* PB[P31]->PB[P33] ON(立上) ?      */
297:         if (port & 0x2) {               /* PB[PC2] ON ? 送信ｽﾀｰﾄ/ｽﾄｯﾌﾟ      */
298:             if (SciStep == 0) SciStep = 1;
299:             else              SciStep = 0;
300:         }
301:         if (SciStep == 0) {            /* 停止中のみ受け付ける             */
302:             if (port & 0x4) {          /* PB[PC5] ON ? ﾎﾞﾚｰﾄ下げ?          */
303:                 if (SciSelect != 0) --SciSelect;
304:             }
305:             else if (port & 0x8) {     /* PB[PF7] ON ? ﾎﾞﾚｰﾄ上げ?          */
306:                 if (BpsTbl[SciSelect+1] != 0) ++SciSelect;
307:             }
308:             Bin2AdecN(dec,BpsTbl[SciSelect],5); /* 表示用データ作成        */
309:             GotoxyMemSet(0,1,dec);
310:         }
311:     }
312: }
```

［リストの説明］　前章に変更を加えた個所を説明します。

２５～２６行：

文字列データの最初と最後を示すコードのシンボル定義です。

３２～３７行：

ＳＣＩ０の送受信割り込み関数で使用する変数の宣言です。

３８～４３行：

ＳＣＩ１の送受信割り込み関数で使用する変数の宣言です。

８２行：

ＳＣＩ０のインタラプトプライオリティを“6”に設定しています。表［２・１－１－３］参照

９９行：

ＳＣＩ０の送受信を割り込み処理にする場合、初期化の段階では送信はディセーブル状態にしておく必要があります。

１１４行：

ＳＣＩ１のインタラプトプライオリティを“6”に設定しています。表［２・１－１－３］参照

１３１行：

ＳＣＩ１の送受信を割り込み処理にする場合、初期化の段階では送信はディセーブル状態にしておく必要があります。

１３６～１４７行：

ＳＣＩ０の送受信割り込みを開始させる関数です。

１３８～１４２行は、文字列送信を割り込みで処理するための準備です。

準備が終了後、最初の１文字を“ＴＤＲ”にセットし、“ＴＤＲＥ＝０”によりＴＳＲに転送後、“ＴＩＥ＝１”により送信データエンプティ割り込み要求を許可しています。

あと残りの文字列データは、送信データエンプティ割り込み処理内で要求が発生するたびに１文字ごと送信をさせます。

１５１～１６２行：

ＳＣＩ１の送受信割り込みを開始させる関数です。ＳＣＩ０の仕様と同じですので省略します。

１６６～１７５行：

ＳＣＩ０の１バイト毎の送信データエンプティ割り込みのハンドラより呼ばれる関数です。

１バイト毎に“ＴＤＲ”にセットし、“ＴＤＲＥ＝０”によりＴＳＲに転送後、内部管理変数を調整して終了しています。

送信するデータがなくなった場合は、送信データエンプティ割り込み要求を禁止して割り込みを止めています。

１７９～１８８行：

ＳＣＩ１の１バイト毎の送信データエンプティ割り込みのハンドラより呼ばれる関数です。

ＳＣＩ０の仕様と同じですので省略します。

１９２～２１０行：

ＳＣＩ０の受信エラーと受信完了の割り込みハンドラより呼ばれる関数です。

１バイト毎の受信完了もしくは受信エラー発生で割り込みが発生します。

シリアルステータスレジスタを監視をおこない、正常時の受信データ処理をします。

受信データがＳＴＸ“￥ｘ０２”の場合、テキストスタートと判断して、内部インデックスをゼロ（０）にします。

受信データがＥＴＸ“￥ｘ０３”の場合、テキスト終了と判断して、受信終了フラグを立てます。

その他の受信データは、内部受信バッファーにセットしていきます。

エラーが発生時は、エラーリセット後内部インデックスカウンタをクリアにしています。

２１４～２３２行：

ＳＣＩ１の受信エラーと受信完了の割り込みハンドラより呼ばれる関数です。

ＳＣＩ０の仕様と同じですので省略します。

２３６～２８６行：

ＳＣＩ０とＳＣＩ１の動作確認用の関数です。

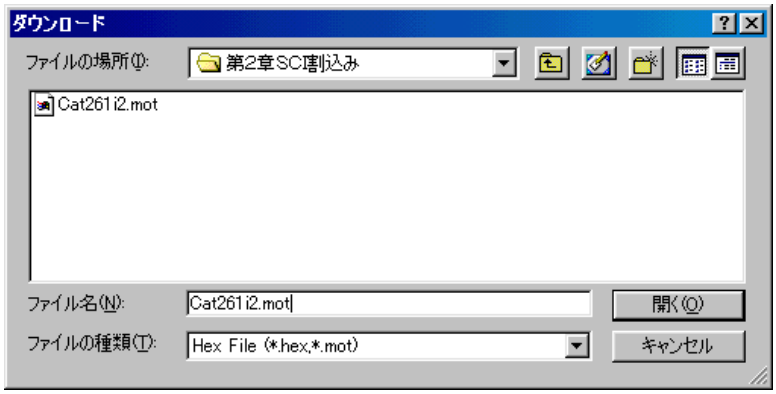
文字列送信後、文字列受信の終了を待ち,終了後ＬＣＤに表示させています。

この動作を、ＰＢ［ＰＣ２］ＯＮか、もしくはエラー発生するまで繰り返します。

無応答判定も、タイマー割り込みを利用した内部ソフトタイマを使用しましたので、ポーリング時に使用していたループソフトタイマを排除しました。

以上の３ファイルの変更で、ＳＣＩ割り込み対応のシステムに変貌したわけです。

ＤＥＦでダウンロード後、プログラム実行をしてみて下さい。



送信データは、“ＩｎｔｅｒｒｕｐｔＳＣ”の文字列を１文字ずつ空けて分けて送信しています。

［ＳＣＩ０＝“Ｉ　ｔ　ｒ　ｕ　ｔ　Ｃ”］　［ＳＣＩ１＝“　ｎ　ｅ　ｒ　ｐ　Ｓ　”］

各チャンネルでの受信データは、ＬＣＤの１行目に表示しますので、正常受信していますと重なり合い文字として読み取れるようになっていると思います。

ここまでの変更では、メロディの割り込み対応は済んでいません。

次章は、メロディの割り込み対応にし、より良いシステムに変貌させたいと思います。

第３章　割込み総合デモ（メロディ）

いよいよ、前章までに築き上げた割り込みシステムを利用した、応用例として割込み総合デモ（メロディ♪♪）のアプリケーション作り上げの最終解説となりました。

リストに沿って説明を進めていきます。
¥評価ボード¥第２部割込み編¥第３章割込み総合デモ（メロディ）¥
にディレクトリの移動をしておいて下さい。

１．総合デモ（メロディ）を割り込み対応にする。

ここで残された問題は、リズム（音の長さ）を調整している部分が、いまだにソフトタイマー（ループ式）を利用していることです。
これがシステムの反応（スイッチ入力反応）を鈍らせている原因です。ここを改善します。

１）プログラムリスト

file　“IntrMelody.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  割込み                                                     */
  4: /*                                                                           */
  5: /*  ＜MOD＞       IntrMelody.c                                               */
  6: /*  ＜役割＞      デモ メロディー関係                                        */
  7: /*  ＜TAB＞       ４タブ編集                                                 */
  8: /*  ＜保守ﾂｰﾙ＞   makefile 参照                                              */
  9: /*  ＜使用ﾊｰﾄﾞ＞  CAT-261-H8S/2612 エーワン(株)                              */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <string.h>
 13: #include    "io261x.h"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*          音階 Hz                                                          */
 17: /*****************************************************************************/
 18: #define d0          262                 /* _ド */
```

```
#define rE          293             /* _ﾚ  */
#define mI          330             /* _ﾐ  */
#define fhA         349             /* _ﾌｧ */
#define sO          392             /* _ｿ  */
#define rA          440             /* _ﾗ  */
#define sI          494             /* _ｼ  */
#define do          523             /* ﾄﾞ  */
#define re          587             /* ﾚ   */
#define mi          659             /* ﾐ   */
#define fha         698             /* ﾌｧ  */
#define so          784             /* ｿ   */
#define ra          880             /* ﾗ   */
#define si          987             /* ｼ   */
#define Do          1047            /* ﾄﾞ  */

#define SCMAX       32              /* 楽譜テーブルの最大数              */
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       MelStep;        /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                       */
        Uchar       MelMode;        /* 演奏ﾓｰﾄﾞ                           */
        Uchar       MelSelect;      /* 自動選曲                           */
        Ushort      MelodyHz;       /* Melody Hz                          */
        Uchar       Music;          /* 自動演奏ｶｳﾝﾀｰ                      */
        Ushort      Doremi[SCMAX];  /* ﾄﾞﾚﾐ音階周波数(Hz)のRAM側           */
        Ushort      Rhythm[SCMAX];  /* ﾘｽﾞﾑ(msec)                         */
const   Ushort      DoremiTbl[] =   /* ﾄﾞﾚﾐ音階周波数(Hz)                 */
{
        do,         /* ﾄﾞ */
        re,         /* ﾚ  */
        mi,         /* ﾐ  */
        fha,        /* ﾌｧ */
        so,         /* ｿ  */
        ra,         /* ﾗ  */
        si,         /* ｼ  */
        Do,         /* ﾄﾞ */
```

```
55:         0
56: };
57: const   Ushort      MusicTbl[3][SCMAX] =    /* 自動演奏の楽譜（最大32ﾏﾃﾞ）   */
58: {                                                           /*  ﾈｺﾌﾝｼﾞｬｯﾀ   */
59:     { ra, so, do, Do, Do, ra, so, do, Do, Do,
60:       ra, so, do, Do, rA, Do, sO, si, si},
61:                                                           /*  ｲﾇﾉｵﾏﾜﾘｻﾝ   */
62:     { mi, do, do, do, mi, do, do, do,fha,fha, mi, mi, re,
63:      fha,fha, mi, mi, re, re, ra, ra, so,fha, mi, re, do},
64:                                                           /*  ｱﾏﾘﾘｽ       */
65:     { so, ra, so, Do, so, ra, so, ra, ra, so, ra, so,fha, mi, re, mi, do, 0},
66: };
67: const   Ushort      RhythmTbl[3][SCMAX] =   /* 自動演奏のリズム(最大32ﾏﾃﾞ) */
68: {                                                           /*  ﾈｺﾌﾝｼﾞｬｯﾀ   */
69:     {250,250,500,500,500,250,250,500,500,500,
70:      250,250,500,500,500,500,500,500,500},
71:                                                           /*  ｲﾇﾉｵﾏﾜﾘｻﾝ   */
72:     {250,250,250,250,250,250,250,250,250,250,250,250,250,1000,
73:      250,250,250,250,250,250,250,250,250,250,250,250,250,1000,},
74:                                                           /*  ｱﾏﾘﾘｽ       */
75:     {500,500,500,500,500,500,1000,500,500,500,500,250,250,250,250,500,500, 0},
76: };
77: const   Uchar       *MelodyTblA[] =     /*  自動演奏の曲名                  */
78: {
79:         ″          ″,
80:         ″ﾈｺﾌﾝｼﾞｬｯﾀ″,
81:         ″ｲﾇﾉｵﾏﾜﾘｻﾝ″,
82:         ″ｱﾏﾘﾘｽ     ″,
83: };
84:
85: /*****************************************************************************/
86: /*      Melody          ﾒﾛﾃﾞｨｰｺﾝﾄﾛｰﾙ                                       */
87: /*****************************************************************************/
88: void    Melody()
89: {
90:     MelodySequence();                   /* ﾒﾛﾃﾞｨｰ操作ｺﾝﾄﾛｰﾙ                 */
```

```
    if (MelMode == 0) ManualMelody();   /* 手動演奏                          */
    else              AutoMelody();     /* 自動演奏                          */
}
/*************************************************************************/
/*      MelodySequence()    ﾒﾛﾃﾞｨｰ操作ｺﾝﾄﾛｰﾙ                                 */
/*************************************************************************/
void    MelodySequence()
{
    Uchar   port;

    port = GetUpPort(1);                /* PB[PA0]->PB[PF7]                  */
    if (port & 0xe) {                   /* PB[PC2]->PB[PF7] ON(立上) ?       */
        Buzzer(0);                      /* Buzer OFF                         */
        if (port & 0x2) {               /* PB[PC2] ON ? ﾓｰﾄﾞ変更             */
            MelStep = 0;                /* 強制停止                          */
            if (MelMode == 0) {         /* 現在手動 ?                        */
                MelMode   = 1;          /* 自動に変更                        */
                MelSelect = 1;          /* 自動選曲                          */
            }
            else {                      /* 現在自動 ?                        */
                MelMode   = 0;          /* 手動に変更                        */
                MelSelect = 0;          /* 自動選曲                          */
            }
        }
        if (MelMode != 0) {             /* 自動 ?                            */
            if (port & 0x4) {           /* PB[PC5] ON ?  選曲                */
                MelStep = 0;            /* 強制停止                          */
                if (++MelSelect > 3) MelSelect = 1;
            }
            if (port & 0x8) {           /* PB[PF7] ON ? 自動演奏開始         */
                if (MelStep == 0) MelStep = 1;      /*  開始                 */
                else              MelStep = 0;      /*  停止                 */
                Music   = 0;
            }
        }
        if (MelMode == 0) GotoxyMemSet(4,1,"M");    /* 手動表示              */
```

```
127:         else                GotoxyMemSet(4,1,"A");     /* 自動表示              */
128:         GotoxyMemSet(7,0,(Uchar *)MelodyTblA[MelSelect]);   /* 曲名表示      */
129:     }
130: }
131: /*****************************************************************************/
132: /*       Ｍｅｍ初期化                                                        */
133: /*****************************************************************************/
134: void    MelMemInitial(void)
135: {
136:     MelStep   = 0;                      /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                   */
137:     MelMode   = 0;                      /* 演奏ﾓｰﾄﾞ                       */
138:     MelSelect = 0;                      /* 自動選曲                       */
139:     MelodyHz  = 0;                      /* Melody Min Hz                  */
140: }
141: /*****************************************************************************/
142: /*       Ｉ／Ｏ初期化                                                        */
143: /*****************************************************************************/
144: void    MelIoInitial(void)
145: {
146:     Buzzer(0);                          /* Buzer OFF                      */
147: }
148: /*****************************************************************************/
149: /*       ManualMelody    手動ﾒﾛﾃﾞｨｰ演奏                                     */
150: /*****************************************************************************/
151: void    ManualMelody()
152: {
153:     Uchar   port;
154:
155:     switch(MelStep) {
156:     case 0:
157:         _strcpyW(Doremi,(Ushort *)DoremiTbl);   /* ﾄﾞﾚﾐ音階周波数(初期準備)*/
158:         MelodyHz  = 0;                  /* Melody Min Hz                  */
159:         MelStep++;
160:         break;
161:     case 1:
162:         if (GetInPort(0) & 0xff) {      /* P40->P47 どれかがＯＮしたか？  */
```

```
163:             port = GetUpPort(0);
164:             if (port & 0x1) {          /* SW[P40] 立上がり ON (ド)          */
165:                 Buzzer(MelodyHz = Doremi[7]);
166:             }
167:             else if (port & 0x2) {     /* SW[P41] 立上がり ON (レ)          */
168:                 Buzzer(MelodyHz = Doremi[6]);
169:             }
170:             else if (port & 0x4) {     /* SW[P42] 立上がり ON (ミ)          */
171:                 Buzzer(MelodyHz = Doremi[5]);
172:             }
173:             else if (port & 0x8) {     /* SW[P43] 立上がり ON (ファ)        */
174:                 Buzzer(MelodyHz = Doremi[4]);
175:             }
176:             else if (port & 0x10) {    /* SW[P44] 立上がり ON (ソ)          */
177:                 Buzzer(MelodyHz = Doremi[3]);
178:             }
179:             else if (port & 0x20) {    /* SW[P45] 立上がり ON (ラ)          */
180:                 Buzzer(MelodyHz = Doremi[2]);
181:             }
182:             else if (port & 0x40) {    /* SW[P46] 立上がり ON (シ)          */
183:                 Buzzer(MelodyHz = Doremi[1]);
184:             }
185:             else if (port & 0x80) {    /* SW[P47] 立上がり ON (ド)          */
186:                 Buzzer(MelodyHz = Doremi[0]);
187:             }
188:         }
189:         else if (MelodyHz != 0) {      /* Buzzer 停止                       */
190:             Buzzer(MelodyHz = 0);
191:         }
192:         break;
193:     }
194: }
195: /*************************************************************************/
196: /*      AutoMelody    自動ﾒﾛﾃﾞｨ演奏                                      */
197: /*************************************************************************/
198: void    AutoMelody()
```

```
199: {
200:     switch(MelStep) {
201:     case 0:
202:         break;
203:     case 1:
204:         _strcpyW(Doremi,(Ushort *)&MusicTbl[MelSelect-1][0]);
205:         _strcpyW(Rhythm,(Ushort *)&RhythmTbl[MelSelect-1][0]);
206:         ++MelStep;
207:         break;
208:     case 2:
209:         MelodyHz = Doremi[Music];       /*  楽譜        */
210:         if (MelodyHz != 0) {            /*  演奏中      */
211:             Buzzer(MelodyHz);
212:             TmStart(TM2,Rhythm[Music]); /* <-リズム開始 */
213:             ++MelStep;
214:         }
215:         else {
216:             Buzzer(0);                  /*  停止        */
217:             Music = 0;
218:             TmStart(TM2,500);           /* <-連続時の間 */
219:             MelStep = 4;
220:         }
221:         break;
222:     case 3:                             /* <-リズム     */
223:         if (TmUpTest(TM2) == ON) {
224:             Music++;
225:             MelStep = 2;
226:         }
227:         break;
228:     case 4:                             /* <-連続時の間 */
229:         if (TmUpTest(TM2) == ON) {
230:             MelStep = 2;
231:         }
232:         break;
233:     }
234: }
```

［リストの説明］　前章に変更を加えた個所を説明します。

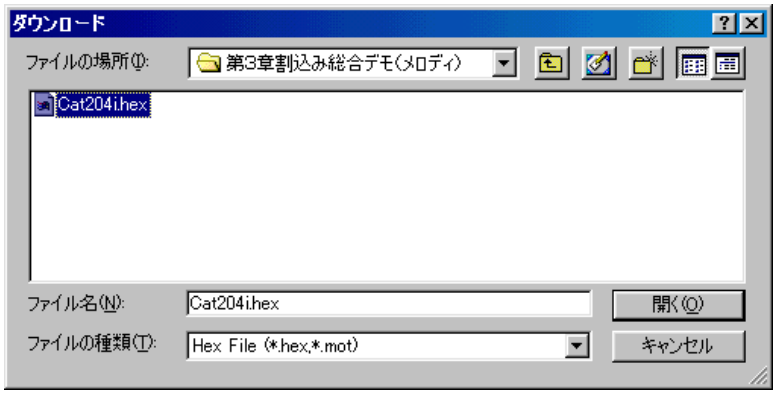
２１２行：　２１８行：　２２３行：　２２９行：

自動演奏のソフトタイマ使用部分を、タイマ割り込みを利用した内部ソフトタイマに変更しました。

これで、通常動作時のソフトタイマ（ループ方式）使用は、全て排除したかたちになりました。

どの程度、動作がスムーズになったか、動かして確認して見ましょう。

ＤＥＦでダウンロード後、プログラム実行をして下さい。



いかがですか？

自動演奏を実行した時にＬＥＤが止まらないでしょう！

（止まるのも味があるのですが、今回の目的とは違います）

それに、いつＰＢ［Ｐｎｎ］を押しても即反応するでしょう！

これでシステムの反応が良くなりました。これが割り込み処理を導入したことによるメリットです。

しかし、このシステムは改造する部分（半音が入っていない等…）が、まだあると思います。

どうぞ改造にチャレンジして、より良いシステムに構築してみて下さい！！　期待しています。

これで、この解説テキストの終了とします。

この内容で満足して頂いたとは思いませんが、皆様のご協力で成長させていきたいと思っています。

メールで頂いたご意見、ご質問を、出来る限り反映させ、より良い解説テキストにしていきたいと考えていますので、ご指導の程よろしくお願い致します。

２００２年　４月　著者