評価ボード

アセンブラ・C 言語による
実践解説テキスト

（BFE204−CAT204　for　LSIC−80）

初版　2002年　3月
改1　2002年　4月
改2　2002年　6月

Rev 1．05（2002／06／25）

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㈱　開発部　長谷川正博
URL: http://www.aone.co.jp
mail: hasegawa@aone.co.jp

［参考文献］

高速８ビットマイクロコントローラ　ＫＬ５Ｃ８０１２ＣＦＰ　ハードウェアマニュアル
川崎マイクロエレクトロニクス㈱

ＬＳＩＣ－８０（Ｖｅｒ３．６）ユーザーズマニュアル
エル・エス・アイ　ジャパン㈱

筋書および概説

本書は、評価ボード（ＢＦＥ２０４　エーワン㈱製）と超小型マイコン（ＣＡＴ２０４　エーワン㈱製）とバグファインダー（ＢＦ３０００　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、評価ボードに付いている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、エル・エス・アイ　ジャパン㈱製のＬＳＩＣ－８０（Ｖｅｒ３．６）を使用しています。

※ＬＳＩ　Ｃ－８０　ｆｏｒ　ＢＦ３０００を使用して、本文中のｋｍｍｍａｋｅ．ｅｘｅを実行する場合には、“ｋｍｍａｋｅ　－ｆ　ＭＡＫＥＤＯＳ”とコマンドを打ってください。

目　　次

ハード構成およびシステム構成 3

第１章　ハード構成 3

第２章　システム構成 4

１．システムブロック図（メロディ♪♪） 4

２．ＣＡＴ２０４のプログラムメモリＭＡＰ 5

３．評価ボードのディップ SW の設定とＩ／Ｏマップ表 5

４．とにかく動かしてみましょう！！ 7

第１部　ポーリング編 11

第１章　スタートアップと MMU 11

１．動かしてみましょう 11

２．プログラムリスト 12

３．保守ツール（ＭａｋｅＦｉｌｅ） 16

第２章　ＬＳＩＣによるスタートアップとMMU 23

１．動かしてみましょう 23

２．プログラムリスト 24

３．保守ツール（ＭａｋｅＦｉｌｅ） 40

第３章　ＰＩＯ 42

１．動かしてみましょう 42

２．プログラムリスト 44

第４章　ＰＩＯの応用（ＬＣＤ） 56

１．動かしてみましょう 56

２．プログラムリスト 57

第５章　タイマ／カウンタ 71

１．動かしてみましょう 72

２．プログラムリスト 73

第６章　ＵＳＡＲＴ 89

１．動かしてみましょう 89

２．プログラムリスト 90

第７章　総合デモ（メロディ） 106

１．動かしてみましょう 106

２．プログラムリスト 107

第２部　割込み編 123

第１章　タイマ割込み 123
１．スタートアップを割り込み用に変更する。 123
２．タイマモジュールを割り込み用に変更する。 129
３．メインコントロール部を割り込み用に変更する。 134

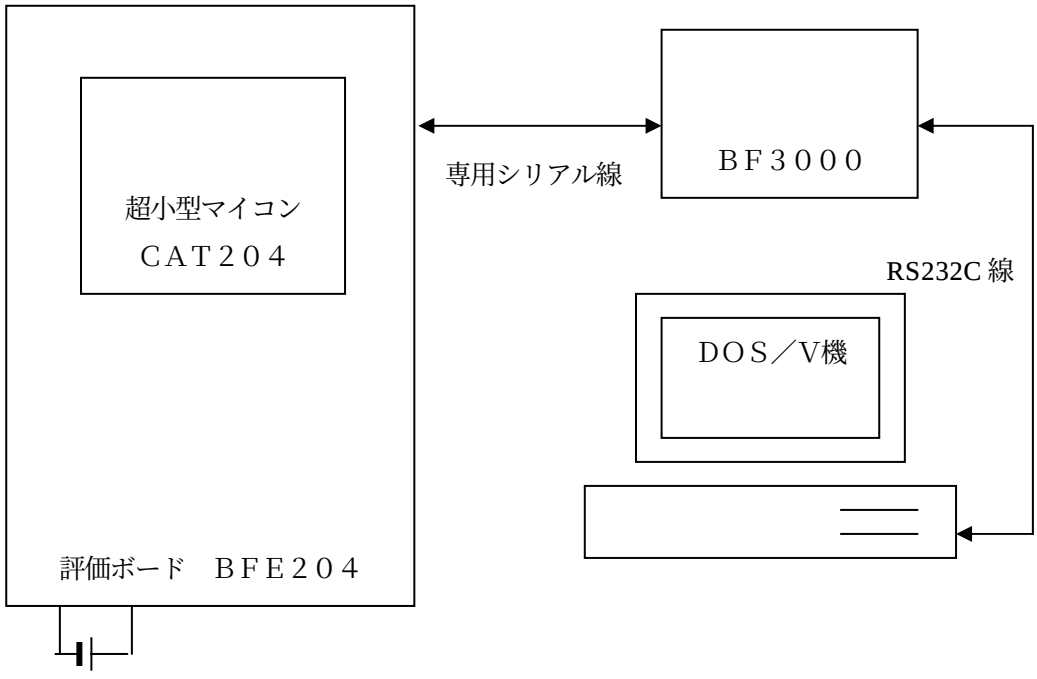
第２章　ＵＳＡＲＴ割込み 145
１．スタートアップを変更する。 145
２．ＵＳＡＲＴモジュールを割り込み用に変更する。 150
３．メインコントロール部の割り込みコントローラを変更する。 158

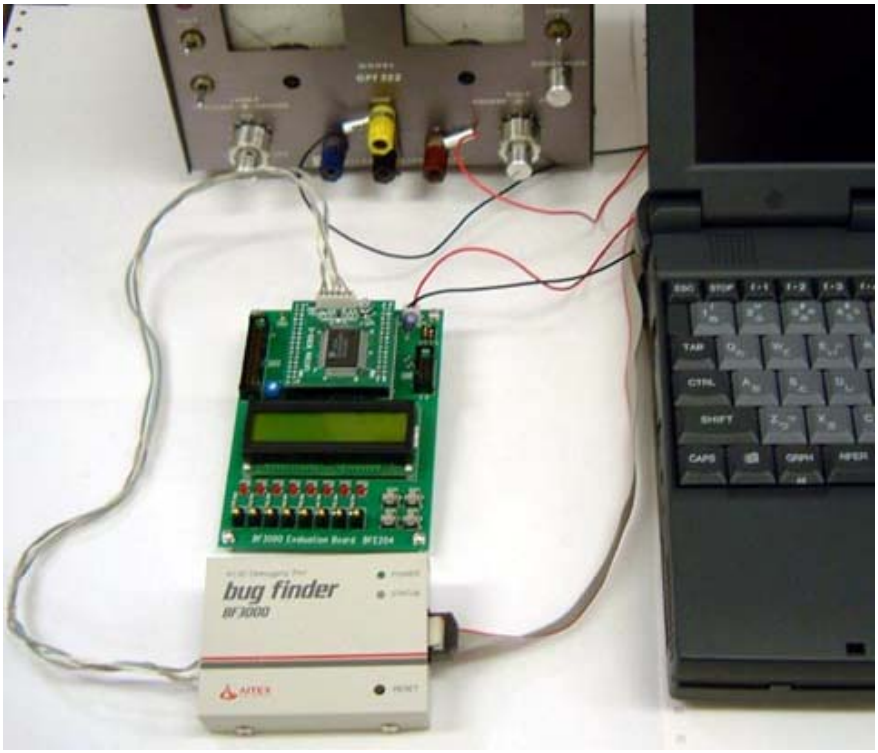
第３章　割込み総合デモ（メロディ） 165
１．総合デモ（メロディ）を割り込み対応にする。 165

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。

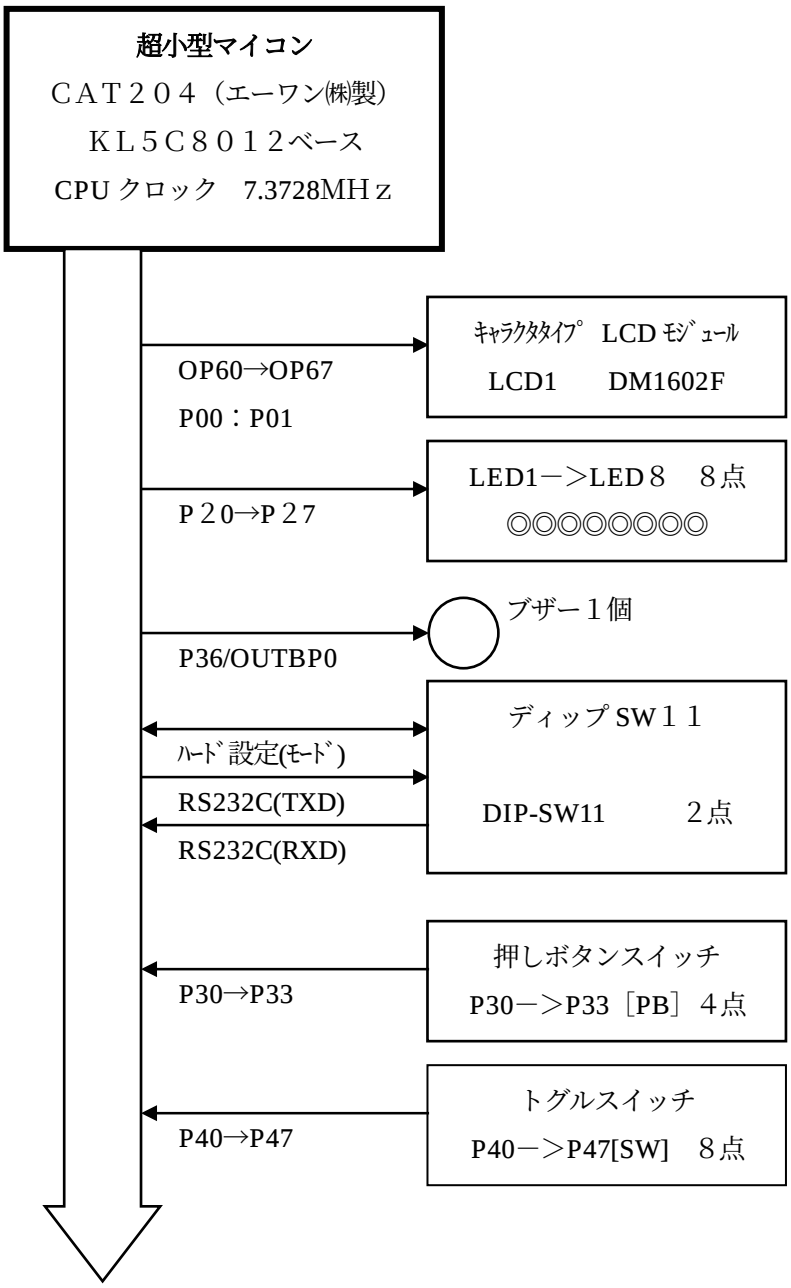
［評価ボード回路図］

“￥評価ボード￥第３部資料￥評価ボード（ＢＦＥ２０４）図面￥＊．ｐｄｆ”参照

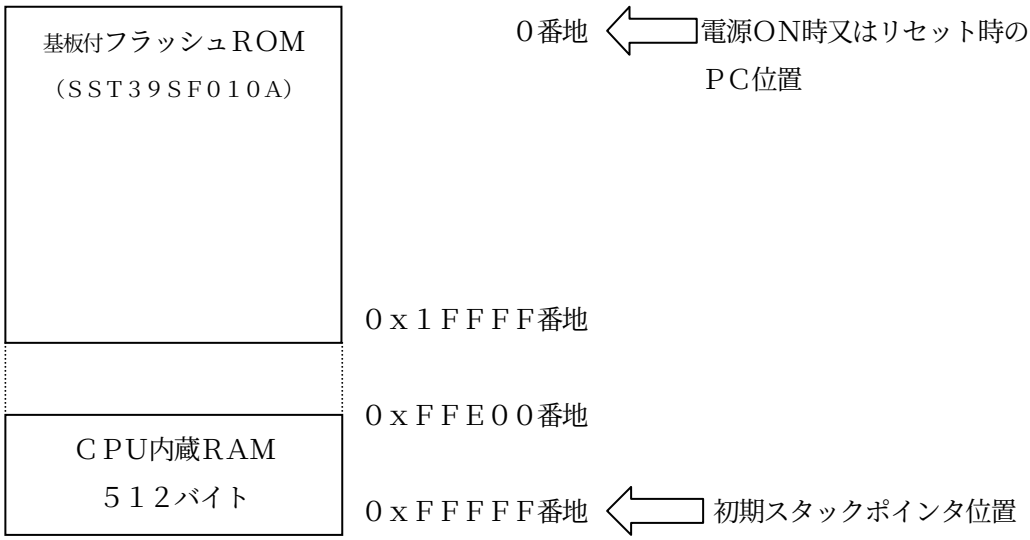
［ＣＡＴ２０４回路図］

“￥評価ボード￥第３部資料￥超小型マイコン（ＣＡＴ２０４）図面￥＊．ｐｄｆ”参照

１．システムブロック図（メロディ♪♪）



２．ＣＡＴ２０４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 TXD,RXD を折り返し	シリアルＩ／Ｏの TXD,RXD 間オープン
ＳＷ１１－２	バグファインダ Boot On Ram モード	ノーマルモード

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
０ｘ６０ （拡張）	PORTE0	CN1-5B	OP60	出力	D0　LCD-module
		1-5A	OP61	出力	D1
		1-4B	OP62	出力	D2
		1-4A	OP63	出力	D3
		1-3B	OP64	出力	D4
		1-3A	OP65	出力	D5
		1-2B	OP66	出力	D6
		1-2A	OP67	出力	D7
０ｘ２Ｃ	PORTA0	CN1-17B	P00	出力	RS　LCD-module
		-17A	P01	出力	E　LCD-module
		-16B	P02		未使用
		-16A	P03		未使用
		-15B	P04		未使用
		-15A	P05		未使用
		-14B	P06		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
０ｘ３０	PORTB0	CN2-12B	P20	出力	LED1
		2-12A	P21	出力	LED2
		2-11B	P22	出力	LED3
		2-11A	P23	出力	LED4
		2-10B	P24	出力	LED5
		2-10A	P25	出力	LED6
		2-9B	P26	出力	LED7
		2-9A	P27	出力	LED8

ブザー関係（タイマ／カウンタＢ　チャネル０　ＯＵＴＢＰ０）				
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
０ｘ２０	ＴＣＮＴB0	CN2-5B	P36/OUTBP0	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
０ｘ３１	PORTB1	CN2-8B	P30	入力	SW12　PB[P30]
		2-8A	P31	入力	SW13　PB[P31]
		2-7B	P32	入力	SW14　PB[P32]
		2-7A	P33	入力	SW15　PB[P33]
		2-6B	P34		未使用
		2-6A	P35		未使用
		2-5B	P36	出力	（ﾌﾞｻﾞｰで使用）
		2-5A	P37		未使用

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
０ｘ３２	PORTB２	CN2-4B	P40	入力	SW16　SW[P40]
		2-4A	P41	入力	SW17　SW[P41]
		2-3B	P42	入力	SW18　SW[P42]
		2-3A	P43	入力	SW19　SW[P43]
		2-2B	P44	入力	SW20　SW[P44]
		2-2A	P45	入力	SW21　SW[P45]
		2-1B	P46	入力	SW22　SW[P46]
		2-1A	P47	入力	SW23　SW[P47]

４．とにかく動かしてみましょう！！

１）パソコン＋ＢＦ３０００＋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ＳＷ１１－２をＯＮにする。

（バグファインダ　Ｂｏｏｔ　Ｏｎ　Ｒａｍモード）

３）評価ボードの電源をオンにする。

４）ＡＢＣｗｉｎを立ち上げる。

５）ＡＢＣｗｉｎの［オプション］－［環境設定］で環境設定をする。

６）ＡＢＣｗｉｎの［オプション］－［ディップスイッチ］でＢＦ３０００のディップスイッチを確認し、ディップスイッチの設定をする。

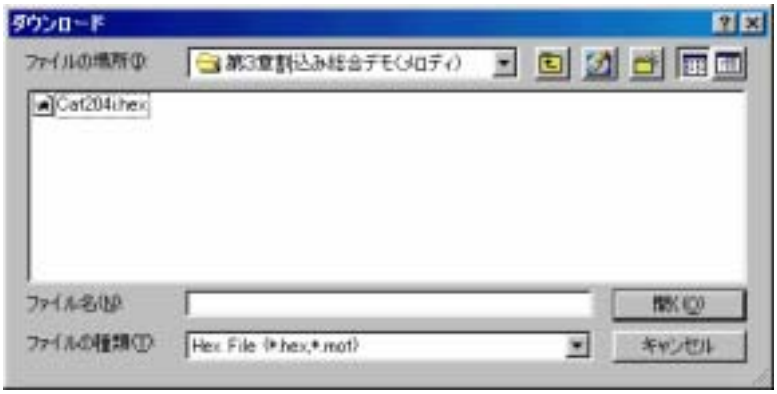
ボーレートを１１５２００ｂｐｓにしますと、［ディップスイッチ］全オフ（上側）になります。

７）ＡＢＣｗｉｎの左下Ｓｔａｒｔをクリックする。

これで、パソコンとＢＦ３０００との通信が可能になります。

８）ＡＢＣｗｉｎ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）\Cat204i.hex**“です。

（ダウンロードの方法は、ＡＢＣｗｉｎのＨＥＬＰを参照）

９）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。

（ＡＢＣｗｉｎのショートＰＢのＧＯをクリック）

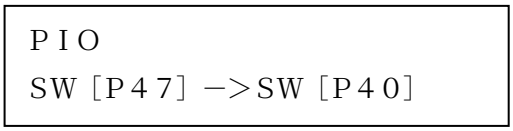
これで、ＬＥＤが何やらパラパラ表示をして、ＬＣＤに文字が表示されるはずです。

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）Ｐ３０を１回押してみてください。

ＰＩＯ動作モード

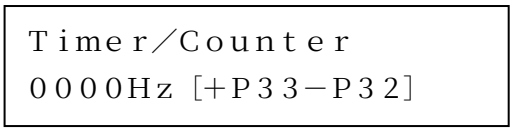


０－４－２図　ＰＩＯ表示

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［Ｐ３０］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［Ｐ３３］でパルス出力の周波数を１００Ｈｚ単位で上げます。
ＰＢ［Ｐ３２］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ＣＮ２－５Ｂをオシロで確認してみて下さい。

ｃ．再度、ＰＢ［Ｐ３０］をもう一回押してみて下さい。

ＵＳＡＲＴ（ＳＩＯ）動作モード

Ｕｓａｒｔ　８，ｎ，１［Ｐ３１］ ０９６００ｂ［＋Ｐ３３－Ｐ３２］

０－４－４図　Ｕｓａｒｔ表示

ＵＳＡＲＴ（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［Ｐ３３］で転送ボーレートを上げることができます。
ＰＢ［Ｐ３２］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をＯＮにして下さい。
ＰＢ［Ｐ３１］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２回に分けて送信します。

１回目“Ｉ　ｔ　ｒ　ｕ　ｔ　Ｃ”　２回目“　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。

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［Ｐ３０］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 Ｐ３１［M］３３［ス］３２［セ］

０－４－5図　Ｍｅｌｏｄｙ表示

ＰＢ［Ｐ３１］でモード変更出来ます。（ｏｎトグル）

Ｐ３１［M］：手動演奏モード

ＳＷ［Ｐ４０］－＞ＳＷ［Ｐ４７］をｏｎ／ｏｆｆしてみて下さい。

Ｐ３１［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［Ｐ３２］で曲を選択して下さい。

ＰＢ［Ｐ３３］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。

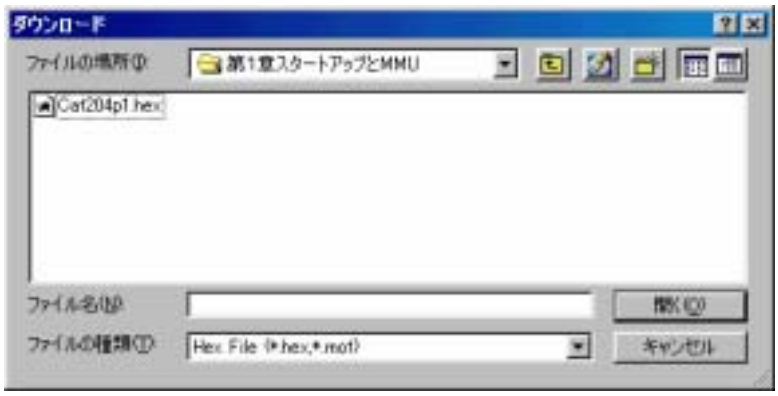
次で説明する第１部ポーリング編は、システム名：“メロディ♪♪”を割込みを一切使用せずに全ポーリング仕様で書いた場合の説明です。
そして第１部が終了しますと、第２部割込み編として割込み処理を取り入れ、よりスムーズなシステムに変更します。
これが、テーマ“メロディ♪♪”になるわけです。

第１部　ポーリング編

第１章　スタートアップと MMU

　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。
ＫＣ８０の場合、MMUもスタートアップ時に設定したほうが望ましいので、一緒に説明します。
また、この章のみ全アセンブラーにて、１本のソースで記述しました。

まずは、ＡＢＣｗｉｎ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 MMU¥



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、“Ｃａｔ２０４ｐ１．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。
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

説明しやすくするため、わざとＥＱＵ等の擬似命令はできる限り使用を制限しました。

file “StartupA.asm”

```
 1: ;/*****************************************************************************/
 2: ;/* <サンプル>  ポーリング                                                    */
 3: ;/*                                                                           */
 4: ;/*     <MOD>     StartupA.asm                                                */
 5: ;/*     <役割>    スタートアップ (CAT204-KL5C8012)                            */
 6: ;/*               main                                                        */
 7: ;/*     <タブ>    ４タブ編集                                                  */
 8: ;/*****************************************************************************/
 9: ;/*****************************************************************************/
10: ;/*          スタートアップ                                                   */
11: ;/*****************************************************************************/
12:              CSEG
13: StartUp_:
14:              ORG     0
15:              DI                  ;* ﾘｾｯﾄで DI になっていますがﾃﾞﾊﾞｯｶｰで"G 0"よう
16:              LD      SP,0xFFFF           ;* ｽﾀｯｸﾎﾟｲﾝﾀの設定
17: ;/*****************************************************************************/
18: ;/*          MMUの初期設定                                                    */
19: ;/*****************************************************************************/
20:              LD      A,0x3e              ;* 物理 0xFFC00 論理 0xFC00 の設定
21:              OUT     (0x6),A             ;* MMU の R4 を使用する
22:              JP      main_
23: ;/*****************************************************************************/
24: ;/*          メイン                                                           */
25: ;/*****************************************************************************/
26:              ORG     0x200
27: main_::
28:              LD      A,0xc0              ;* SCR1     外部ﾒﾓﾘ 0Wait 外部 I/O 1Wait
29:              OUT     (0x3B),A
30:              LD      A,0x3               ;* PIOB P0 全出力設定 LED 点灯のため
31:              OUT     (0x33),A
```

```
32:             LD      D,1                 ;* LED      表示パターン
33: loop::
34:             LD      A,D
35:             CPL                         ;* PORT     0=点灯のため
36:             OUT     (0x30),A            ;* PORT     出力（LED 点灯）
37:             LD      B,20                ;* 20ms     Wait
38: wait20ms::
39:             CALL    Wait1ms
40:             DJNZ    wait20ms
41:             RLC     D
42:             JR      loop
43: ;/*****************************************************************************/
44: ;/*     Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                    */
45: ;/*****************************************************************************/
46: Wait1ms::
47:             PUSH    HL
48:             LD      HL,1228             ;* 1228*6=7372cyc    */
49: W01:
50:             DEC     HL                  ;* cyc = 1
51:             LD      A,L                 ;*      = 1
52:             OR      H                   ;*      = 1
53:             JP      NZ,W01              ;*      = 3
54:             POP     HL                  ;*     += 6
55:             RET
56: ;
57:             END
```

［リストの説明］

１～１１行：

コメントです。

１２行：

“ＣＳＥＧ”は、リンカにこれからのプログラムはコードセグネントだと教えるための擬似命令です。

１３行：

ラベルです。

１４行：

“ＯＲＧ　０”は、ここからのオフセットアドレスを指定する擬似命令です。

リンク時に、＃ＣＯＤＥでコードセグネントの先頭アドレスを指定しますので、そのアドレスのオフセットアドレスになるわけです。

１５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ます**。

命令“ＤＩ”の意味は割込み禁止です。このサンプルでは割込み未使用なため不要ですが、プログラムミスでＥＩ（割込み許可状態）になってしまい暴走し、デバッカで“Ｇ　０”などした場合に都合が良いためただ入れてあります。

まあ定石だと思って下さい。

なお電源ON時およびリセット時は、ＣＰＵ自身“ＤＩ”状態です。

１６行：

スタックポインタ（ＳＰ）の初期値設定です。

これは絶対必要で、通常ＲＡＭの最終番地を指定するのが標準になっています。

ただ、ＳＰ動作は必ず－１して動作するため、この例ですと０ｘＦＦＦＦ番地は使用しないことになります。

フルにＲＡＭを使用したい場合は、“ＬＤ　ＳＰ，０”になるわけですが、私個人の意見としては直感的でない数字のためこの方法は使用しないようにしています。

２０，２１行：

ＫＣ８０特有のMMU設定の部分です。

詳細はハードウェアマニュアルに記載されていますので、ここではこの２行の説明をします。

このサンプルのようにMMUを頻繁に使用せず、初期化１回設定ですむようなシステムの場合は、ＲＡＭ領域設定にＲ４を使用するのが定石になっています。

なぜかと言いますと、MMUのＢＲ４は０ｘＦ０固定になっており、物理アドレスベースが０ｘＦ００００になりますので都合が良い事になります。

ＣＡＴ２０４の場合、物理アドレスで０ｘＦＦＥ００～０ｘＦＦＦＦＦ番地がＲＡＭになっていますので、ＢＢＲ４の論理アドレス境界値の設定のみで済みます。

ただし、論理アドレス境界値の最小単位は０ｘ４００ですので、０ｘＦＥ００は０ｘＦＣ００にな

ります。

［アドレス境界値計算式］

論理アドレス境界設定値＝（論理アドレス境界値÷０ｘ４００）－１

０ｘ３ｅ＝（０ｘＦＣ００÷０ｘ４００）－１

ＢＢＲ４の上位２ビットは、ゼロ固定になっていますので設定値“０ｘ３ｅ”が得られたことになります。

ＢＢＲ４のＩ／Ｏアドレスは、０ｘ６番地ですので、この２行の記述になります。

２２行：

“ＪＰ　ｍａｉｎ＿”メイン処理へのジャンプ命令になります。

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！！

２３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、ｋｍｍａｋｅ．ｅｘｅ（ＬＳＩＣ－８０パッケージに添付）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、“ｋｍｍａｋｅ”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の仕様は、ＬＳＩＣ－８０のマニュアルに記載されていますが、簡単ではありますが、ここでちょっと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: #
 2: # Makefile  for KC80   LSIC-80  by AONE
 3: #
 4: #    (kmmake にて実行する)
 5: #
 6: # ４タブ編集
 7: #
 8: # 関係拡張子の登録（下記以降での拡張子記述は必ず小文字の事）
 9: #
10: .SUFFIXES: .sof .asm .asz .psm .psz .mac .c .h .hex
11:
12: # プロジェクト名(SAMPLE.xxx)
13: PROJ   = Cat204p1
14:
15: # ライブラリー格納ディレクトリ名
16: LIB    = s:\lsij\lsic80\lib
17:
18: # ユザー作成のヘッダーファイル名
19: HEDS   =
20:
21: # ユザー作成のオブジェクトファイル名
22: OBJS   = StartupA.sof
23:
24: # コンパイル／アセンブラ　コマンド名
```

```
25: CC80   = LCC80
26: ASM80  = rz80
27:
28: # ＬＣＣ８０　コンパイル　スイッチ説明
29: #      -z80 Ｚ８０で実行可能な命令が生成されます
30: #      -r   3ﾊﾞｲﾄJMPを2ﾊﾞｲﾄBRAに変更
31: #      -c   コンパイル時にリンクは実行しません
32: #      -cn  コメントのネスト許可
33: #      -cv  クラスＤＡＴＡに配置された変数を宣言順番に生成します
34: #      -g1  行番号情報とグローバル変数情報を生成する
35: #      -j1  エラーメッセージを日本語で表示します
36: #      -v1  実行しているコマンドを表示します
37: #      -wi  宣言をしていない関数を使用する場合の警告メッセージを抑えます
38: CFLAGS = -z80 -r -c -cn -cv -g1 -j1 -v1 -wi
39:
40: # ｒｚ８０　アセンブリ　スイッチ説明
41: #      -g  行番号情報を生成する
42: #      -ｒ Ｃと同様な数値記述を許す
43: ASMFLAGS = -g -r
44:
45: # リンカーコマンド名
46: LINK   = knil
47:
48: # リンカー指定　注意　各行の先頭は、タブ（ＴＡＢ）送りする。
49: $(PROJ).hex : $(OBJS)
50:   $(LINK) @${
51:           -v
52:           -tH
53:           -c
54:           $(PROJ).hex
55:           #CODE=0000 #DATA=FE00
56:           -KC80 -BB4FE00,FFE00
57:           $(OBJS)
58:           -f$(LIB)\clibz
59:           -f$(LIB)\flibz
60:           -f$(LIB)\romlibz
```

```
61:             -f$(LIB)\knjlibz
62:             -m$(PROJ).MAP
63:             -n$(PROJ).LIN
64:             }
65: #
66: # 各リンカースイッチの説明
67: #
68: # -v                # 処理の進行状況を表示します
69: # -tH               # 拡張インテルＨＥＸを指定
70: # -c                # セグメントの重複を検査します
71: # $(PROJ).hex       # ＨＥＸファイル名の指定
72: # #CODE=0000        #  ＣＯＤＥ／ＤＡＴＡセグメントの先頭アドレスの指定
73: # #DATA=FE00
74: # -KC80 -BB4FE00,FFE00
75: #                   # ＫＣ８０のMMU初期状態の指定（$$Rx_init_value_を作成）
76: # $(OBJS)           #  オブジェクト指定（スタートアップは必ずこの位置）
77: # -f$(LIB)\clibz                          # ライブラリ指定
78: # -f$(LIB)\flibz
79: # -f$(LIB)\romlibz
80: # -f$(LIB)\knjlibz
81: # -m$(PROJ).MAP                           # マップファイル名の指定
82: # -n$(PROJ).LIN                           # ラインファイル名の指定
83:
84: # 全てのオブジェクト調査
85: # all: $(OBJS)
86:
87: # オブジェクトとヘッダーの調査
88: $(OBJS) : $(HEDS)
89:
90: # コンパイル実行
91: .c.sof:
92:   $(CC80) $(CFLAGS) $<
93: .asm.sof:
94:   $(ASM80) $(ASMFLAGS) $<
```

［ＭａｋｅＦｉｌｅの説明］

１～９行：

コメント（注釈）です。頭に“＃”を付けるとコメントになるルールです。

１０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。**

１３行：

“PROJ = Cat204p1”は、便宜上プロジェクト（テーマ）名を環境変数“PROJ”に登録しています。

環境変数は予約語と重ならない限り自由に指定できます。

１６行：

“LIB = s:\lsij\lsic80\lib”も便宜上、ライブラリーの置いてあるディレクトリ名を環境変数“LIB”に登録しています。

注意としては、当社の開発環境では“ｓ：”ドライブに開発用ソフトウェアをインストールしていますので、“ｓ：￥～”になっています。

皆様の開発環境は様々だと思います。

（重要！）ここのディレクトリ名は、各自ＬＳＩＣ－８０をインストールした場所に変更して下さい。

１９行：

“HEDS =”は、プロジェクトでヘッダーファイル（定義文をまとめた）を作成した場合、便宜上この環境変数に登録しておきます。

後にでてきますが、ヘッダーファイルを変更した場合、再コンパイル／アセンブリが必要ですのでここに登録しておきます。

２２行：

“OBJS = StartupA.sof”は、プロジェクトを作成するにあたり、作成した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）名を便宜上この環境変数に登録しておきます。

ＬＳＩＣ－８０では、“＊．ｓｏｆ”になります。

２５行：

“CC80 = LCC80”は、コンパイラのコマンド名を登録しています。

２６行：

“ASM80 = rz80”は、アセンブラのコマンド名を登録しています。

３８行：

“CFLAGS = -z80 -r -c -cn -cv -g1 -j1 -v1 –wi”は、コンパイルを実行する時のスイッチを登録しています。

４３行：

“ ASMFLAGS = -g –r ”は、アセンブラを実行する時のスイッチを登録しています。

４６行：

“ LINK = knil ”は、リンカーコマンド名の登録です。

４９行：

“ $(PROJ).hex : $(OBJS) ”は、前に登録した（プロジェクト名）．ｈｅｘ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）．ｈｅｘのほうが古い（昔にできた）場合、次行のリンクコマンド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。”

５０行：

リンクコマンドの発行とリンクスイッチのリストです。

“ @ ”は、ファイルからコマンド行を読み込む指示です。

“ $ {.....} ”は、“ {”と“}”の間のリンクスイッチリストを展開後“ｍａｋｅ．ｉ”ファイルに作成されます。

注意事項として、各行の先頭は、<TAB>送りをしてください。

そうしないと正しく“ｍａｋｅ．ｉ”が作成されません。

５１～６３行：

リンクコマンドスイッチを指定しています。

８８行：

“ $(OBJS):$(HEDS) ”は,オブジェクトとヘッダーファイルの作成時期の関係を調べます

オブジェクトのほうが古い（昔にできた）場合、次行のコンパイル又はアセンブリを実行します。

９１行：

“ .c.sof: ”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。

９３行：

“ .asm.sof: ”は、Ａｓｍソースファイル“．ａｓｍ”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。

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１３、１９、２２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）ｋｍｍａｋｅ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。

“ＳｔａｒｔｕｐＡ．ａｓｍ”をエディタで開き

```
36:             OUT     (0x30),A            ;* PORT     出力（LED 点灯）
37:             LD      B,20                ;* 20ms     Wait
38: wait20ms::
```

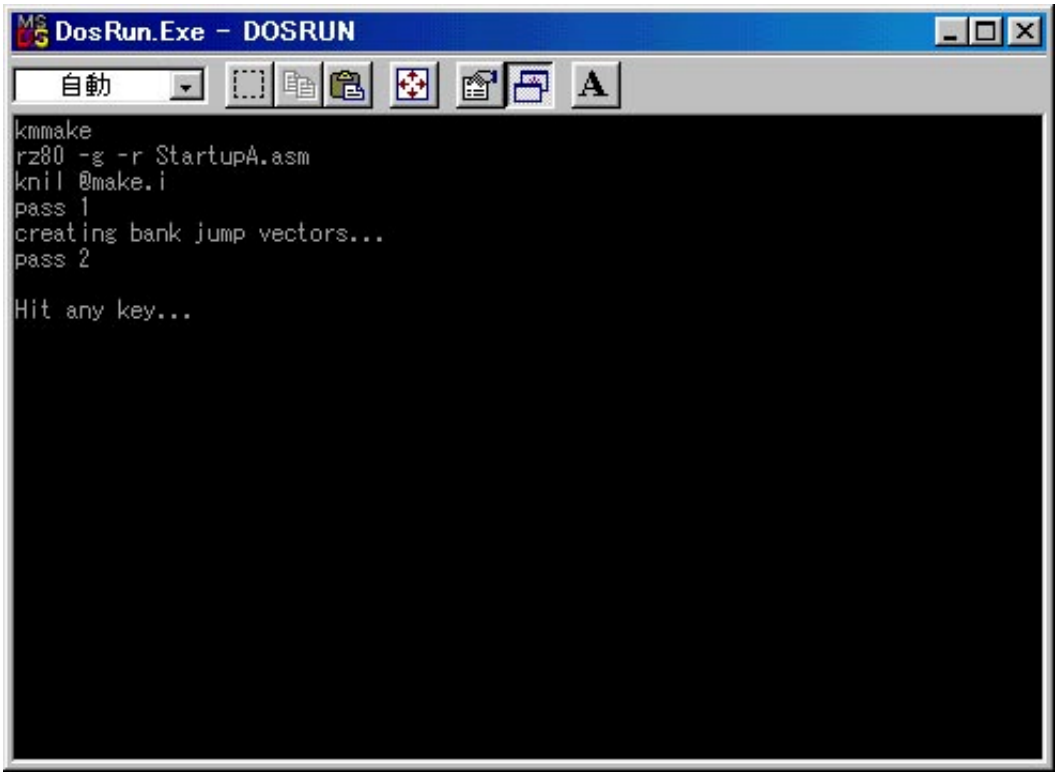
３７行目の“２０”を“１００”に変更してみてください。

```
37:             LD      B,100               ;* 100ms    Wait
```

変更後、保存終了して下さい。

保存終了が済みましたら、“ｋｍｍａｋｅ”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のご利益により、プロジェクト名：Ｃａｔ２０４ｐ１．ｈｅｘ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。

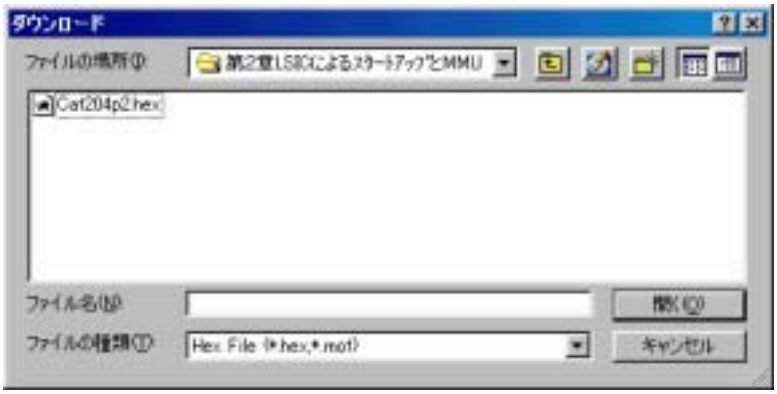
プログラムデバック作業は、簡単に言えばこの繰り返しです。

思ったように動かない場合は、デバッカ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“ｋｍｍａｋｅ”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　ＬＳＩＣによるスタートアップとMMU

　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、ＡＢＣｗｉｎ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ＬＳＩＣによるスタートアップとMMU￥

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、**“Ｃａｔ２０４ｐ２．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。

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３ファイルに分割しました。

１）“ＳｔａｒｔｕｐＢ．ａｓｍ”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。

２）“Ｃａｔ２０４ｐ２．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。

３）“Ｐ＿Ｐｉｏ１．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３ファイル作成しました。

１）“ＣＡＴ２０４．ｈ”は、ＣＡＴ２０４（超小型マイコン）の使用するＩ／Ｏ番地をシンボル定義でまとめたファイルです。

２）“ＫＣ８０１２ＩＯ．ｈ”は、ＣＡＴ２０４（超小型マイコン）が使用しているＣＰＵ（ＫＬ５Ｃ８０１２）の内部Ｉ／Ｏをシンボル定義でまとめたファイルです。“ＣＡＴ２０４．ｈ”が使用しています。

３）“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）スタートアップ（プログラム）

今回はリストを読みやすくするためＥＱＵ等の擬似命令を使用しました。

file　“StartupB.asm”

```
 1: ;/*****************************************************************************/
 2: ;/*      <MOD>      StartupB.asm                                              */
 3: ;/*      <役割>     スタートアップ（ポーリング専用 CAT204-KL5C8012)           */
 4: ;/*                 リンクの都合上、先頭に指定することが絶対条件              */
 5: ;/*                 $$Rx_init_value_は、knil（リンク）で作成されます。        */
 6: ;/*      <TAB>      ４タブ編集                                               */
 7: ;/*****************************************************************************/
 8: STACK       EQU     0xFFFF          ; ｽﾀｯｸﾎﾞﾄﾑ
 9: BBR1        EQU     0x0             ; KL5C8012(MMU) BBR1
10: BR1         EQU     0x1             ;               BR1
11: BBR2        EQU     0x2             ;               BBR2
12: BR2         EQU     0x3             ;               BR2
13: BBR3        EQU     0x4             ;               BBR3
14: BR3         EQU     0x5             ;               BR3
15: BBR4        EQU     0x6             ;               BBR4
16: BR4         EQU     0x7             ;               BR4
17: ;/*****************************************************************************/
18: ;/*          スタートアップ                                                   */
19: ;/*****************************************************************************/
20:             CSEG
21: StartUp_:
22:             ORG     0
23:             DI
24:             LD      SP,STACK
25: ;/*****************************************************************************/
26: ;/*          MMUの初期設定                                                    */
27: ;/*****************************************************************************/
28:             LD      A,LOW  $$R1_init_value_##
29:             OUT     (BBR1),A
30:             LD      A,HIGH $$R1_init_value_##
31:             OUT     (BR1),A
```

```
32:             LD      A,LOW  $$R2_init_value_##
33:             OUT     (BBR2),A
34:             LD      A,HIGH $$R2_init_value_##
35:             OUT     (BR2),A
36:             LD      A,LOW  $$R3_init_value_##
37:             OUT     (BBR3),A
38:             LD      A,HIGH $$R3_init_value_##
39:             OUT     (BR3),A
40:             LD      A,LOW  $$R4_init_value_##
41:             OUT     (BBR4),A
42:             JP      main_##
43: ;/*****************************************************************************/
44: ;/*          終了（Cソースへのエントリー）                                    */
45: ;/*****************************************************************************/
46:             ORG     0x200
47:             END
```

［リストの説明］

1～7行：

コメントです。

４行のコメントに記載してありますが、このモジュールをアセンブリして出来あがったオブジェクト“ＳｔａｒｔｕｐＢ．ｓｏｆ”は、リンク時に必ず１番目で指定して下さい。

8～１６行：

使用している数値をシンボル化する EQU 定義文です。

前章と比べてみて下さい。

２８～41行：

MMU に対する初期設定文です。

文中にでてくる“ $$Rn_init_value_ ”は、リンク（ｋｎｉｌ）の“－Ｂ”スイッチで指定する物理アドレスに対する論理アドレス指定と使用するＲｎの指定で作成されます。（ｎ＝１～４）

実際に出力される“ $$Rn_init_value_ ”の数値は、リンクで作成されるファイル“__ｍｍｕｉｎｉｔ．ａｓｚ”を読めば確認できます。

なお、“ $$Rn_init_value_ ”の最後に“ ## ”が書いてありますが、これはアセンブラ（ｒｚ８０）の擬似命令で外部参照（ＥＸＴＲＮ）を意味しています。

４２行：

Ｃ言語記述の“ｍａｉｎ（）”へのエントリーです。

“ｍａｉｎ__”と最後に“__”（アンダーバー）がついていますが、これは C 言語で宣言したシンボルに対して付加されます。

リンクスイッチで“__”を取ることもできますが、ＣとＡＳＭの区別のためあったほうが良いかもしれません。（皆様の自由です）

４６行：

“ ORG　0x200 ”は、C 言語記述のオブジェクトを離すためにいれました。

メモリがもったいないと思う方は削除しても構いません。

４７行：

“ END ”は、アセンブラソースの最後を知らせる擬似命令です。

必ず入れて下さい。

＊2）メイン処理（プログラム）

file “Cat204p2.c”

```
/*****************************************************************************/
/*                                                                           */
/*  <サンプル>  ポーリング                                                   */
/*                                                                           */
/*  <MOD>       Cat204p2.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-204-KL5C8012 エーワン(株)                                 */
/*                                                                           */
/*****************************************************************************/
#include    <machine.H>
#include    "CAT204.H"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      変数宣言                                                             */
/*****************************************************************************/
        Uchar       Shift;              /* shift ﾊﾟﾀｰﾝ                       */
/*****************************************************************************/
/*      main()                                                               */
/*****************************************************************************/
void    main(void)
{
    outp(SCR1,0xc0);                    /* SYS    ExtMem 0wait ExtIO 1wait  */
    SoftWait1ms(20);                    /* Power On Wait(20ms) 安定待ち      */
    MemInitial();                       /* ﾒﾓﾘ系初期化                       */
    IoInitial();                        /* I/O 系初期化                      */
    while(1) {
        SoftWait1ms(20);                /* 20msWait                          */
        RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ      */
        SigOutput();                    /* Signal Output Process(LED 点灯)   */
    }
}
```

```
34: /*****************************************************************************/
35: /*      Ｍｅｍ初期化                                                         */
36: /*****************************************************************************/
37: void    MemInitial(void)
38: {
39:     Shift = 0;                          /* Led Disp Patan Initial          */
40:
41:     PioMemInitial();                    /* PIO    Mem 初期化               */
42: }
43: /*****************************************************************************/
44: /*      Ｉ／Ｏ初期化                                                         */
45: /*****************************************************************************/
46: void    IoInitial(void)
47: {
48:     PioIoInitial();                     /* PIO    I/O 初期化               */
49: }
50: /*****************************************************************************/
51: /*      RunRun()        CPU 走行表示                                         */
52: /*****************************************************************************/
53: void    RunRun()
54: {
55:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
56:     PutOutPort(Shift,'=');
57: }
58: /*****************************************************************************/
59: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                     */
60: /*****************************************************************************/
61: void    SoftWait1ms(Ushort ms)
62: {
63:     while(ms-- != 0) {
64:         Wait1ms();
65:     }
66: }
67: /*****************************************************************************/
68: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                   */
69: /*****************************************************************************/
```

```
70: void    Wait1ms()
71: {
72:     _asm_("\n           PUSH    HL              \n");
73:     _asm_("\n           LD      HL,1228         \n");   /* 1228*6=7372cyc   */
74:     _asm_("\n W01:                              \n");
75:     _asm_("\n           DEC     HL              \n");   /* cyc = 1          */
76:     _asm_("\n           LD      A,L             \n");   /*       = 1        */
77:     _asm_("\n           OR      H               \n");   /*       = 1        */
78:     _asm_("\n           JP      NZ,W01          \n");   /*       = 3        */
79:     _asm_("\n           POP     HL              \n");   /*     += 6         */
80: }
```

［リストの説明］

１～１０行：

コメントです。

１２行：

“#include <machine.H>”は、ＬＳＩＣ標準添付の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。

ポートの入出力関数（inp()､outp()等）を使用する場合、記述します。

１３～１４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。

１８行：

内部使用する変数の宣言文です。

“Uchar”は、“Cat204.h”でマクロ宣言してあるため、“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。

２２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。

２４行：

ＣＰＵ（ＫＬ５Ｃ８０１２以後“ＫＣ８０”と略す）特有の手続き（ポート出力）です。

外部バス・ウェイト・コントロールの設定が目的です。**図［2－2－1］参照**

ＣＡＴ２０４は、外部メモリ（０～７ＦＦＦＦ）＝０ｗａｉｔ、外部メモリ（８０００～ＦＦＤＦＦ）＝なし、外部Ｉ／Ｏ＝１ｗａｉｔの設定ですので、“１１”になり、設定値“０ｘｃ０”となります。**ＫＣ８０を使用する場合、必要です。**

```
outp(SCR1,0xc0);                    /* SYS    ExtMem 0wait ExtIO 1wait  */
```

（ＳＣＲ１＝ＣＡＴ２０４．ＨでＩ／Ｏアドレスのシンボル宣言済みです）

つまり、この行までのプログラムはデフォルトの１ｗａｉｔの状態で動作し、この行以降から０ｗａｉｔ状態で動作します。

システムコントロールレジスタ（ＳＣＲ１）		
ﾋﾞｯﾄ	名称	機能
7 6	外部バス・ ウェイトコントロール	外部メモリ（0～7ＦＦＦＦ）　外部メモリ（８０００～ＦＦＤＦＦ）　外部Ｉ／Ｏ ００：　１ウェイト　１ウェイト　２ウェイト ０１：　１ウェイト＊　１ウェイト＊　２ウェイト＊ １０：　１ウェイト　０ウェイト　１ウェイト **１１：　０ウェイト　０ウェイト　１ウェイト** ＊ワイド・ライト・ストローブ・オプション
5	―	
4	―	
3	端子８５の機能	１：ＫＣ８２のＮＭＩ＿入力として機能します。 ０：Ｐ０３として機能します。
2	端子９２，９６の機能	１：ＢＲＥＱ＿，ＢＡＣＫ＿として機能します。 ０：Ｐ４０，Ｐ４４として機能します。
1	端子７１の機能	１：ＨＡＬＴ＿を出力します。 ０：Ｐ１７として機能します。
0	端子７２の機能	１：Ｍ１＿を出力します。 ０：Ｐ１６として機能します。

図[２-２-１]　システムコントロールレジスタ１

２５行：

電源ＯＮ時に周辺Ｉ／Ｏが安定するまで待つソフトタイマーです。

ＣＡＴ２０４ボードを使用する場合は不要ですが、私めの定石／癖になっています。

無視して下さい。

２６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。

２７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。

２８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。

２９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）

３０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。

３１行：

ｏｕｔバッファーをポート出力する関数を呼んでいます。

３２行：

前に説明した“while(1)”の終わりを示す記号です。

３３行：

“main()”関数の終わりを示す記号です。

この２３行目から始まり、３３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。

３７～４２行：

メモリ初期化の関数です。

４６～４９行：

Ｉ／Ｏ初期化の関数です。

５３～５７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。

６１～６６行：

ｍｓｅｃ単位で指定するソフトタイマ関数です。

７０～８０行：

約１ｍｅｃのソフトタイマです。

ＣＰＵクロック＝７３７２８００Ｈｚですので、１０００Ｈｚ（１ｍｓ）で割りますとマシンサイクル＝７３７２サイクルになります。

１ループ６サイクルですので、７３７２÷６＝１２２８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 “P_Pio1.c” LED 表示（動作目視用）

```
  1: /*************************************************************************/
  2: /*                                                                       */
  3: /*  ＜サンプル＞  ポーリング                                             */
  4: /*                                                                       */
  5: /*  ＜MOD＞       P_Pio1.c                                               */
  6: /*  ＜役割＞      PIO 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-KL5C8012 エーワン(株)                          */
 10: /*                                                                       */
 11: /*************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.H"
 14: #include    "DemoCtl.h"
 15: /*************************************************************************/
 16: /*      変数宣言                                                         */
 17: /*************************************************************************/
 18:         Uchar       OutPort;            /* Out  Port  Buffer            */
 19: /*************************************************************************/
 20: /*      Ｍｅｍ初期化                                                     */
 21: /*************************************************************************/
 22: void    PioMemInitial(void)
 23: {
 24:     OutPort = 0;                        /* OUT  Port  初期化            */
 25: }
 26: /*************************************************************************/
 27: /*      Ｉ／Ｏ初期化                                                     */
 28: /*************************************************************************/
 29: void    PioIoInitial(void)
 30: {
 31:     outp(PDIRBX,0x3);                   /* PIOB P0  全出力              */
 32:                                         /*      P1  未使用 全入力       */
 33:                                         /*      P2  未使用 全入力       */
```

```
34: }
35: /*************************************************************************/
36: /*      SigOutput   Signal Output Process(LED 出力)                      */
37: /*************************************************************************/
38: void    SigOutput()
39: {
40:     outp(PORTB0,~OutPort);              /* 0=点灯 1=消灯 のため          */
41: }
42: /*************************************************************************/
43: /*      OutPortPut  出力バッファーにセット                               */
44: /*************************************************************************/
45: void    PutOutPort(Uchar patan,Uchar log)
46: {
47:     if (log == '=')      OutPort  = patan;
48:     else if (log == '|') OutPort |= patan;
49:     else if (log == '&') OutPort &= patan;
50: }
```

［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 “CAT204.h”

```
 1: /*****************************************************************************/
 2: /*       マクロ宣言                                                          */
 3: /*****************************************************************************/
 4: #define Uchar   unsigned char
 5: #define Ushort  unsigned short
 6: #define Ulong   unsigned long
 7:
 8: /*****************************************************************************/
 9: /*       ＫＬ５Ｃ８０Ａ１２内部（Ｉ／Ｏ番地）                               */
10: /*****************************************************************************/
11: #include "KC8012IO.h"
12:
13: /*****************************************************************************/
14: /*            ＣＡＴ－２０４（拡張Ｉ／Ｏ番地）                               */
15: /*****************************************************************************/
16: #define PORTE0  0x60                    /* PIOE 6 出力 PORT                  */
17: #define PORTE1  0x70                    /*      7  "  PORT                  */
```

［リストの説明］

4～6行：

“unsigned”長い予約語のため、このように予約語をマクロ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。

１1行：

別に用意したＫＬ５Ｃ８０１２の内部Ｉ／Ｏをシンボル化したヘッダファイルをインクルードします。

１6～１7行：

ＣＡＴ２０４の拡張Ｉ／Ｏアドレスをシンボル化しました。

*2）ヘッダーファイル（2）

file “KC8012IO.h”

```
/*****************************************************************************/
/*        ＫＬ５Ｃ８０Ａ１２内部（Ｉ／Ｏ番地）                              */
/*****************************************************************************/
#define BBR1      0x0             /* MMU   1 境界                          */
#define BR1       0x1             /*         ベース                        */
#define BBR2      0x2             /*       2 境界                          */
#define BR2       0x3             /*         ベース                        */
#define BBR3      0x4             /*       3 境界                          */
#define BR3       0x5             /*         ベース                        */
#define BBR4      0x6             /*       4 境界                          */
#define BR4       0x7             /*         ベース                        */
#define TCNTB0    0x20            /* TIMB  0 CNT                           */
#define TCWDB0    0x21            /*       0 CWORD/STAT                    */
#define TCNTB1    0x22            /*       1 CNT                           */
#define TCWDB1    0x23            /*       1 CWORD/STAT                    */
#define TCNTB2    0x24            /*       2 CNT                           */
#define TCWDB2    0x25            /*       2 CWORD/STAT                    */
#define TCNTA0    0x28            /* TIMA  0 CNT                           */
#define TCWDA0    0x29            /*       0 CWORD/STAT                    */
#define TCNTA1    0x2A            /*       1 CNT                           */
#define TCWDA1    0x2B            /*       1 CWORD/STAT                    */
#define PORTA0    0x2C            /* PIOA  0 PORT                          */
#define PDIRA0    0x2D            /*       0 方向                          */
#define PORTA1    0x2E            /*       1 PORT                          */
#define PDIRA1    0x2F            /*       1 方向                          */
#define PORTB0    0x30            /* PIOB  0 PORT                          */
#define PORTB1    0x31            /*       1 PORT                          */
#define PORTB2    0x32            /*       2 PORT                          */
#define PDIRBX    0x33            /*       ｺﾝﾄﾛｰﾙ+方向                     */
#define LERL      0x34            /* 割込  X LERL                          */
#define LERH      0x35            /*       X LERH                          */
#define PGRL      LERL            /*       X PGRL                          */
#define PGRH      LERH            /*       X PGRH                          */
```

```
34: #define ISRL      LERL              /*        X ISRL                */
35: #define ISRH      LERH              /*        X ISRH                */
36: #define IMRL      0x36              /*        X IMRL                */
37: #define IMRH      0x37              /*        X IMRH                */
38: #define IVR       IMRH              /*        X IVR                 */
39: #define K51DAT0   0x38              /* USART 0 TXD/RXD               */
40: #define K51COM0   0x39              /*        0 MODE/STAT            */
41: #define SCR0      0x3A              /* SYS    0 SCR0                 */
42: #define SCR1      0x3B              /*        1 SCR1                 */
```

［リストの説明］

ＫＬ５Ｃ８０Ａ１２内部のＩ／Ｏアドレスをシンボル化しました。

*3）ヘッダーファイル（3）

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();
/*      P_Pio1.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。

特に**ＬＳＩＣの場合はプロトタイプ宣言が重要です。**

なぜかと言いますと関数の第１引数は、データサイズによりレジスタが変わるからです。

プロトタイプ宣言をせずに、他モジュールの関数を使用した場合、第１引数はデフォルトで“ｉｎｔ”（２バイト）と判断され、コンパイラは、ＨＬレジに引数をセットします。

もとの関数の第１引数が“ｃｈａｒ”系で宣言をしてあった場合、その関数はＡレジを見ていますので、正しく動作しません。

このようなトラブルを避けるためにも**プロトタイプ宣言を必ずすることをお勧めします。**

	第１引数	第２引数	第３引数
１バイト（ｃｈａｒ）	Ａ	Ｅ	Ｃ
２バイト（ｓｈｏｒｔ）	ＨＬ	ＤＥ	ＢＣ

図［２－２－２］　ＬＳＩＣにおける引数とレジスタの関係

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

```
12: #  プロジェクト名(SAMPLE.xxx)
13: PROJ     = Cat204p2
・
18: #  ユザー作成のヘッダーファイル名
19: HEDS     =   CAT204.h KC8012IO.h DemoCtl.h
・
21: #  ユザー作成のオブジェクトファイル名
22: OBJS     = StartupB.sof Cat204p2.sof P_Pio1.sof
```

どうです前章で説明したように、１３，１９，２２行を変更しただけです。

くどいようですが１６行目のライブラリー格納ディレクトリ名は、**各自環境に変更**して下さい。

２）ｋｍｍａｋｅ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

“Ｃａｔ２０４ｐ２．ｃ”をエディタで開き

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

２９行目の“２０”を“１００”に変更してみてください。

```
29:         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、“ｋｍｍａｋｅ”を実行してみて下さい。

ｋｍｍａｋｅで新しく出来た“Ｃａｔ２０４ｐ２．ｈｅｘ”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、ＬＣＤ表示［第４章で説明します］）です。

まずは、ＡＢＣｗｉｎ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、**“Ｃａｔ２０４ｐ３．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。

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）**Ｐ３０**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［Ｐ２７］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［Ｐ３０］**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。

これだけの仕様ですが、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）＜－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、３ファイルの構成になっています。

“ＳｔａｒｔｕｐＢ．ａｓｍ”　前章のまま使用したので解説を省略します。

“Ｐ＿Ｐｉｏ２．ｃ”　ＰＩＯ関係をまとめました。

“Ｃａｔ２０４ｐ３．ｃ”　メインコントロール部です。

１）ＰＩＯ関係

file　“P_Pio2.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  ポーリング                                                 */
  4: /*                                                                           */
  5: /*  ＜MOD＞       P_Pio2.c                                                   */
  6: /*  ＜役割＞      PIO 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-KL5C8012 エーワン(株)                               */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.H"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*      変数宣言                                                             */
 17: /*****************************************************************************/
 18:         Uchar       InPort[2];          /* IN   Port  現 Buffer             */
 19:         Uchar       UpPort[2];          /* IN   Port  立上り Buffer         */
 20:         Uchar       InBack[2];          /* IN   Port  一ヶ前 Buffer         */
 21:         Uchar       In20ms[2];          /* IN   Port  20ms 前 Buffe         */
 22:         Uchar       InNows[2];          /* IN   Port  生 Buffer             */
 23:         Uchar       OutPort;            /* Out  Port  Buffer                */
 24: /*****************************************************************************/
 25: /*      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:     outp(PORTA0,0);                     /* PIOA P0  使用   D0=LCD-RS 出力   */
42:     outp(PDIRA0,0x3);                   /*                 D1=LCD-E  出力   */
43:     outp(PDIRA1,0);                     /*      P1  未使用 全入力           */
44:     outp(PORTB0,0);                     /* PIOB P0  全出力                  */
45:                                         /*      P1  全入力［ＰＢ］D0-D3     */
46:     outp(PDIRBX,0x3);                   /*      P2  全入力[ＳＷ]            */
47:     outp(PORTE0,0);                     /* PIOE P0  拡張   全出力 OFF       */
48:     outp(PORTE1,0);                     /*      P1  拡張   全出力 OFF       */
49: }
50: /*************************************************************************/
51: /*      SigInput  Signal Input Process(チャタ取り+状態検出+立上り検出)      */
52: /*************************************************************************/
53: void    SigInput()
54: {
55:     InNows[0] = ~inp(PORTB2);           /* IN生 負論理 SW[P40->P47]         */
56:     InNows[1] = (~inp(PORTB1) & 0xf);   /* IN生 負論理 PB[P30->P33]         */
57:     InPort[0] = In20ms[0] & InNows[0];  /* チャタ取り+状態検出[P40->P47]   */
58:     InPort[1] = In20ms[1] & InNows[1];  /*        "            [P30->P33]    */
59:     UpPort[0] = (InPort[0] ^ InBack[0]) & InPort[0]; /* 立上検出[P40->P47]  */
60:     UpPort[1] = (InPort[1] ^ InBack[1]) & InPort[1]; /*     "   [P30->P33]  */
61:     In20ms[0] = InNows[0];              /* 20ms前Buffe記憶[P40->P47]        */
62:     In20ms[1] = InNows[1];              /*       "         [P30->P33]       */
```

```
63:     InBack[0] = InPort[0];              /* １ヶ前記憶[P40->P47]            */
64:     InBack[1] = InPort[1];              /*       "   [P30->P33]            */
65: }
66: /*************************************************************************/
67: /*      PioDemo()  ＰＩＯデモ                                            */
68: /*************************************************************************/
69: void    PioDemo()
70: {
71:     if (InPort[0] & 0x1)  OutPort = OutPort |   0x80;          /* SW[P40]  */
72:     else                  OutPort = OutPort & ~(0x80);
73:     if (InPort[0] & 0x2)  OutPort = OutPort |   0x40;          /* SW[P41]  */
74:     else                  OutPort = OutPort & ~(0x40);
75:     if (InPort[0] & 0x4)  OutPort = OutPort |   0x20;          /* SW[P42]  */
76:     else                  OutPort = OutPort & ~(0x20);
77:     if (InPort[0] & 0x8)  OutPort = OutPort |   0x10;          /* SW[P43]  */
78:     else                  OutPort = OutPort & ~(0x10);
79:     if (InPort[0] & 0x10) OutPort = OutPort |   0x8;           /* SW[P44]  */
80:     else                  OutPort = OutPort & ~(0x8);
81:     if (InPort[0] & 0x20) OutPort = OutPort |   0x4;           /* SW[P45]  */
82:     else                  OutPort = OutPort & ~(0x4);
83:     if (InPort[0] & 0x40) OutPort = OutPort |   0x2;           /* SW[P46]  */
84:     else                  OutPort = OutPort & ~(0x2);
85:     if (InPort[0] & 0x80) OutPort = OutPort |   0x1;           /* SW[P47]  */
86:     else                  OutPort = OutPort & ~(0x1);
87: }
88: /*************************************************************************/
89: /*      GetInPort() InPort[x]の読み取り                                  */
90: /*************************************************************************/
91: Uchar   GetInPort(Uchar port)
92: {
93:     return(InPort[port]);
94: }
95: /*************************************************************************/
96: /*      GetUpPort() UpPort[x]の読み取り                                  */
97: /*************************************************************************/
98: Uchar   GetUpPort(Uchar port)
```

```
 99: {
100:     return(UpPort[port]);
101: }
102: /*************************************************************************/
103: /*      SigOutput   Signal Output Process(LED 出力)                      */
104: /*************************************************************************/
105: void    SigOutput()
106: {
107:     outp(PORTB0,~OutPort);              /* 0=点灯 1=消灯 のため          */
108: }
109: /*************************************************************************/
110: /*      OutPortPut  出力バッファーにセット                               */
111: /*************************************************************************/
112: void    PutOutPort(Uchar patan,Uchar log)
113: {
114:     if (log == '=')      OutPort  = patan;
115:     else if (log == '|') OutPort |= patan;
116:     else if (log == '&') OutPort &= patan;
117: }
```

［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。

ＰＩＯを使用するためには、まず入出力の方向を設定する必要があります。

ポートA方向制御レジスタ（ＰＤＩＲAn）		
ﾋﾞｯﾄ	名称	機能
7	PnEN「7」	１：ポートｎのビット７を出力に設定 ０：ポートｎのビット７を入力に設定
6	PnEN［6］	１：ポートｎのビット６を出力に設定 ０：ポートｎのビット６を入力に設定
5	PnEN［5］	１：ポートｎのビット５を出力に設定 ０：ポートｎのビット５を入力に設定
4	PnEN［4］	１：ポートｎのビット４を出力に設定 ０：ポートｎのビット４を入力に設定
3	PnEN［3］	１：ポートｎのビット３を出力に設定 ０：ポートｎのビット３を入力に設定
2	PnEN［2］	１：ポートｎのビット２を出力に設定 ０：ポートｎのビット２を入力に設定
1	PnEN［1］	１：ポートｎのビット１を出力に設定 ０：ポートｎのビット１を入力に設定
0	PnEN［0］	１：ポートｎのビット０を出力に設定 ０：ポートｎのビット０を入力に設定

図［3－2－1］　ポートA方向設定レジスタ　ｎ＝０，１

Ｉ／Ｏアドレス	ブロック名	ライト／リード時	シンボル
２CH	パラレル ポートＡ	ポート０	ＰＯＲＴＡ０
２DH		ポート０の方向制御レジスタ	ＰＤＩＲＡ０
２EH		ポート１	ＰＯＲＴＡ１
２FH		ポート１の方向制御レジスタ	ＰＤＩＲＡ１

図［3－2－2］　ポートAのＩ／Ｏマップ

[４１～４３行]

ポートを出力に指定する前にゼロ（Ｌｏｗ）にしておきます。

ポートＡのＰｏｒｔ０のＰ００とＰ０１を出力に指定しました。図［３－２－１］を参照

ポートＡのＰｏｒｔ１は、未使用ですので、８ビット入力に指定します。

ポートB方向制御レジスタ（ＰＤＩＲＢＸ）		
ﾋﾞｯﾄ	名称	機能
7	－	0
6	－	0
5	P2H	１：ポート２上位４ビットを出力に指定 ０：ポート２上位４ビットを入力に指定
4	P2L	１：ポート２下位４ビットを出力に指定 ０：ポート２下位４ビットを入力に指定
3	P1H	１：ポート１上位４ビットを出力に指定 ０：ポート１上位４ビットを入力に指定
2	P1L	１：ポート１下位４ビットを出力に指定 ０：ポート１下位４ビットを入力に指定
1	P0H	１：ポート０上位４ビットを出力に指定 ０：ポート０上位４ビットを入力に指定
0	P0L	１：ポート０下位４ビットを出力に指定 ０：ポート０下位４ビットを入力に指定

図［3－2－3］　ポートB方向制御レジスタ

Ｉ／Ｏアドレス	ブロック名	ライト／リード時	シンボル
３０Ｈ	パラレル ポートＢ	ポート０	ＰＯＲＴＢ０
３１Ｈ		ポート１	ＰＯＲＴＢ１
３２Ｈ		ポート２	ＰＯＲＴＢ２
３３Ｈ		コントロール／方向制御レジスタ	ＰＤＩＲＢＸ

図［3－2－4］　ポートBのＩ／Ｏマップ

Ｉ／Ｏアドレス	ブロック名	ライト	シンボル
６０Ｈ	拡張ポート 出力専用	ポート６　ＯＰ６ｘ	ＰＯＲＴＥ０
７３Ｈ		ポート７　ＯＰ７ｘ	ＰＯＲＴＥ１

図［3－2－5］　拡張ポートのＩ／Ｏマップ

[４４～４８行]

ポートＢのＰｏｒｔ０は、全ビットをＬＥＤに使用しますので、出力に指定し、Ｐｏｒｔ０をゼロ（Ｌｏｗ）にしておきます。

ポートＢのＰｏｒｔ１は、下位４ビットをＰＢに使用しますので、入力に指定します。

未使用の上位４ビットは、入力にしておきます。

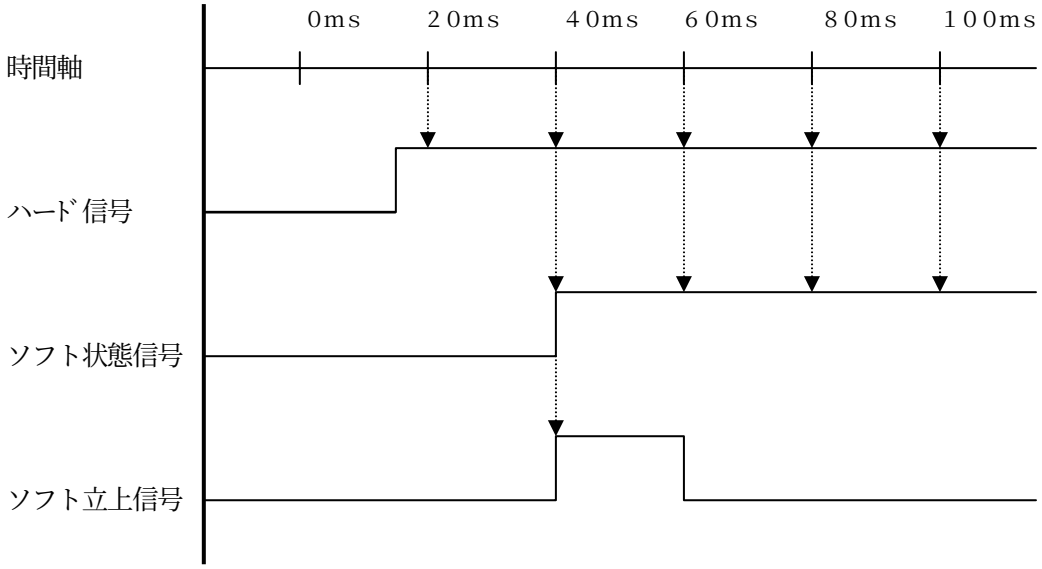
ＣＡＴ２０４の拡張２ポートは出力固定ですので、２ポートをゼロ（Ｌｏｗ）にしておきます。

５３～６５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［Ｐ３３－＞Ｐ３０］
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。

６９～８７行：

ＳＷ［Ｐ４０－＞Ｐ４７］のスイッチをオンすると、ＬＥＤ［Ｐ２０－＞Ｐ２７］を対角線上に点灯させるデモ関数です。

９１～９４行：

ＳＷおよびＰＢのソフト状態信号を取得する関数です。

９８～１０１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。

１０５～１０８行：

出力バッファ“OutPort”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。

１１２～１１７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 “Cat204p3.c”

```
/*****************************************************************************/
/*                                                                           */
/*  <サンプル>  ポーリング                                                   */
/*                                                                           */
/*  <MOD>       Cat204p3.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-204-KL5C8012 エーワン(株)                                 */
/*                                                                           */
/*****************************************************************************/
#include    <machine.H>
#include    "CAT204.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)
{
    outp(SCR1,0xc0);                    /* SYS    ExtMem 0wait ExtIO 1wait  */
    SoftWait1ms(20);                    /* Power On Wait(20ms) 安定待ち      */
    MemInitial();                       /* ﾒﾓﾘ系初期化                       */
    IoInitial();                        /* I/O 系初期化                      */
    while(1) {
        SigInput();                     /* Signal Input  Process             */
        SoftWait1ms(20);                /*       ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り      */
        ModeCntrol();                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                        */
        SigOutput();                    /* Signal Output Process(LED 点灯)   */
```

```
34:     }
35: }
36: /*************************************************************************/
37: /*      Ｍｅｍ初期化                                                     */
38: /*************************************************************************/
39: void    MemInitial(void)
40: {
41:     ModeStep = 0;                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ              */
42:     Shift = 0;                      /* Led Disp Patan Initial          */
43:
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: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                       */
55: /*************************************************************************/
56: void    ModeCntrol()
57: {
58:     if (GetUpPort(1) & 0x1) {       /* PB[P30] ON?(立上)               */
59:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST    */
60:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ     */
61:     }
62:     switch(ModeStep) {
63:     case 0:
64:         ModeStep++;
65:         break;
66:     case 1:
67:         RunRun();                   /* ｼﾌﾄLED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ       */
68:         break;
69:     case 10:                        /*  PIO   TEST                     */
```

```
70:         ModeStep++;
71:         break;
72:     case 11:
73:         PioDemo();
74:         break;
75:     }
76: }
77:
78: /*************************************************************************/
79: /*      RunRun()        CPU 走行表示                                     */
80: /*************************************************************************/
81: void    RunRun()
82: {
83:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
84:     PutOutPort(Shift,'=');
85: }
86: /*************************************************************************/
87: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                 */
88: /*************************************************************************/
89: void    SoftWait1ms(Ushort ms)
90: {
91:     while(ms-- != 0) {
92:         Wait1ms();
93:     }
94: }
95: /*************************************************************************/
96: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait               */
97: /*************************************************************************/
98: void    Wait1ms()
99: {
100:    _asm_("\n           PUSH    HL              \n");
101:    _asm_("\n           LD      HL,1228         \n");   /* 1228*6=7372cyc   */
102:    _asm_("\n W01:                              \n");
103:    _asm_("\n           DEC     HL              \n");   /* cyc = 1          */
104:    _asm_("\n           LD      A,L             \n");   /*     = 1          */
105:    _asm_("\n           OR      H               \n");   /*     = 1          */
```

```
106:    _asm_("\n          JP      NZ,W01          \n");   /*     = 3          */
107:    _asm_("\n          POP     HL              \n");   /*    += 6          */
108: }
```

［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。

３０行：

“Ｐ＿Ｐｉｏ２．ｃ”で作成した、ＳＷとＰＢの入力関数を呼んでいます。

３２行：

モード制御する関数を呼んでいます。

４１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。

４４行：

“Ｐ＿Ｐｉｏ２．ｃ”で使用する変数を初期化する関数を呼んでいます。

５１行：

ＰＩＯのＩ／Ｏ初期化する関数を呼んでいます。

５６～７６行：

モード制御の関数です。

ＰＢ［Ｐ３０］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。

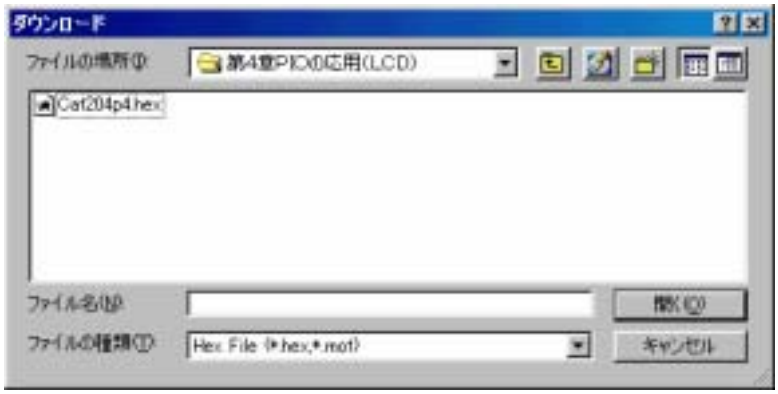
保守ツールについては、もうご理解したと思いますので、この章以降説明は省略します。

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、“ｋｍｍａｋｅ”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、ＡＢＣｗｉｎ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、“Ｃａｔ２０４ｐ４．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？

仕様は、前章と同じですので、ＰＢ［Ｐ３０］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、４ファイルの構成になっています。

“ＳｔａｒｔｕｐＢ．ａｓｍ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｐｉｏ２．ｃ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｌｃｄ．ｃ”	追加したＬＣＤコントロールのモジュールです。
“Ｃａｔ２０４ｐ４．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 “P_Lcd.c”

```
 1: /*************************************************************************/
 2: /*                                                                       */
 3: /*  ＜サンプル＞  ポーリング                                             */
 4: /*                                                                       */
 5: /*  ＜MOD＞       P_Lcd.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-204-KL5C8012 エーワン(株)                            */
10: /*                                                                       */
11: /*************************************************************************/
12: #include    <machine.H>
13: #include    "CAT204.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              */
19: #define EMDCD       0x6             /*          ｴﾝﾄﾘｰﾓｰﾄﾞｾｯﾄ              */
```

```
20: #define FUKCD       0x38                /*          ﾌｧﾝｸｼｮﾝｾｯﾄ              */
21: #define DISON       0xc                 /*          表示ｵﾝ                  */
22: #define DISOFF      0x8                 /*          表示ｵﾌ                  */
23: /*************************************************************************/
24: /*      変数宣言                                                         */
25: /*************************************************************************/
26:         Uchar       LcdBuf[2][16];      /* LCD  表示  Buffer             */
27:         Uchar       LcdReq;             /*      表示  要求ﾌﾗｸﾞ             */
28: /*************************************************************************/
29: /*      Ｍｅｍ初期化                                                     */
30: /*************************************************************************/
31: void    LcdMemInitial(void)
32: {
33:     memset(LcdBuf[0],0x20,16);          /* LCD  表示  Buffer             */
34:     memset(LcdBuf[1],0x20,16);
35:     LcdReq = ON;                        /*      表示  要求ﾌﾗｸﾞ             */
36: }
37: /*************************************************************************/
38: /*      LcdInitial()    LCD イニシャル                                   */
39: /*************************************************************************/
40: void    LcdIoInitial()
41: {
42:     SoftWait1ms(45);                    /* Power On Wait 45ms            */
43:     LcdCmd(FUKCD);                      /* LCD ｺﾏﾝﾄﾞ ﾌｧﾝｸｼｮﾝｾｯﾄ(1)       */
44:     SoftWait1ms(5);                     /*          5ms Wait             */
45:     LcdCmd(FUKCD);                      /* LCD ｺﾏﾝﾄﾞ ﾌｧﾝｸｼｮﾝｾｯﾄ(2)       */
46:     SoftWait10us(10);                   /*          100us Wait           */
47:     LcdCmd(FUKCD);                      /* LCD ｺﾏﾝﾄﾞ ﾌｧﾝｸｼｮﾝｾｯﾄ(3)       */
48:     LcdCmd(FUKCD);                      /* LCD ｺﾏﾝﾄﾞ ﾌｧﾝｸｼｮﾝｾｯﾄ(4)       */
49:     LcdDispOn();                        /* LCD      表示ｵﾝ              */
50:     LcdDispClear();                     /* LCD      表示ｸﾘｱ             */
51:     LcdCmd(EMDCD);                      /* LCD ｺﾏﾝﾄﾞ ｴﾝﾄﾘｰﾓｰﾄﾞｾｯﾄ        */
52:     SoftWait10us(4);                    /*          40us Wait            */
53: }
54: /*************************************************************************/
55: /*      AllLcdDisp()        全画面表示                                   */
```

```
56: /*************************************************************************/
57: void    AllLcdDisp()
58: {
59:     if (LcdReq == ON) {                 /* 表示要求                          */
60:         LcdReq = OFF;                   /*    "     OFF にするのはココだけ   */
61:         LcdDispOff();
62:         GotoxyDisp(0,0,LcdBuf[0]);      /* 1 行目                            */
63:         GotoxyDisp(0,1,LcdBuf[1]);      /* 2 行目                            */
64:         LcdDispOn();
65:     }
66: }
67: /*************************************************************************/
68: /*      GotoxyMemSet()      画面表示バッファーにセット                       */
69: /*************************************************************************/
70: void    GotoxyMemSet(Uchar x,Uchar y,Uchar *str)
71: {
72:     Uchar   *ptr;
73:
74:     ptr = &LcdBuf[y][x];
75:     while(*str != 0) {*ptr++ = *str++;}
76:     LcdReq = ON;                        /* 表示要求 ON にするのはココだけ    */
77: }
78: /*************************************************************************/
79: /*      GotoxyDisp()        カーソル移動＋表示                               */
80: /*************************************************************************/
81: void    GotoxyDisp(Uchar x,Uchar y,Uchar *str)
82: {
83:     Gotoxy(x,y);                        /* カーソル移動                      */
84:     while(*str != 0) {
85:         LcdPutch(*str++);               /* １文字表示                        */
86:     }
87: }
88: /*************************************************************************/
89: /*      Gotoxy()            カーソル移動                                     */
90: /*************************************************************************/
91: void    Gotoxy(Uchar x,Uchar y)
```

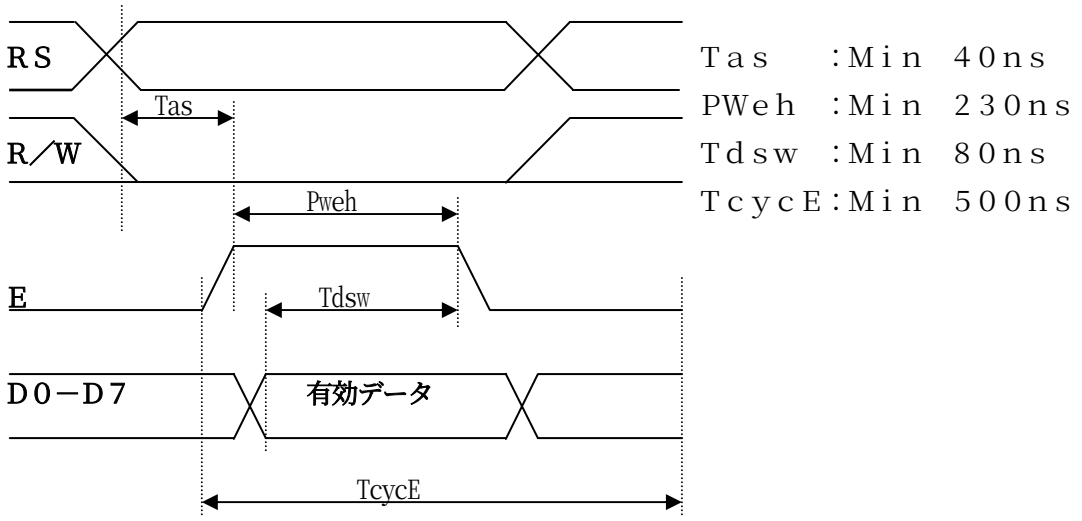
```
 92: {
 93:     Uchar   ramadr;
 94:
 95:     if (y == 0) ramadr = 0;             /* DDRAM ｱﾄﾞﾚｽ計算                  */
 96:     else        ramadr = 0x40;
 97:     ramadr += x;
 98:     LcdCmd(ramadr | 0x80);              /* LCD ｺﾏﾝﾄﾞ DDRAM ｱﾄﾞﾚｽｾｯﾄ          */
 99:     SoftWait10us(4);                    /*          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:     outp(PORTA0,inp(PORTA0) |   0x1);   /* LCD  RS     ON                  */
```

```
128:      if (data < 0x20) data = 0x20;
129:      LcdCmd(data);
130:      outp(PORTA0,inp(PORTA0) & ~(0x1));  /* LCD  RS      OFF                */
131:      SoftWait10us(4);                    /*          40us Wait              */
132: }
133: /*************************************************************************/
134: /*       LcdCmd()     LCD command OUT                                     */
135: /*************************************************************************/
136: void     LcdCmd(Uchar cmd)
137: {
138:      outp(PORTE0,cmd);                   /* LCD  Data                      */
139:      outp(PORTA0,inp(PORTA0) |   0x2);   /*       E      ON                */
140:      outp(PORTA0,inp(PORTA0) & ~(0x2));  /*       E      OFF               */
141: }
```

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

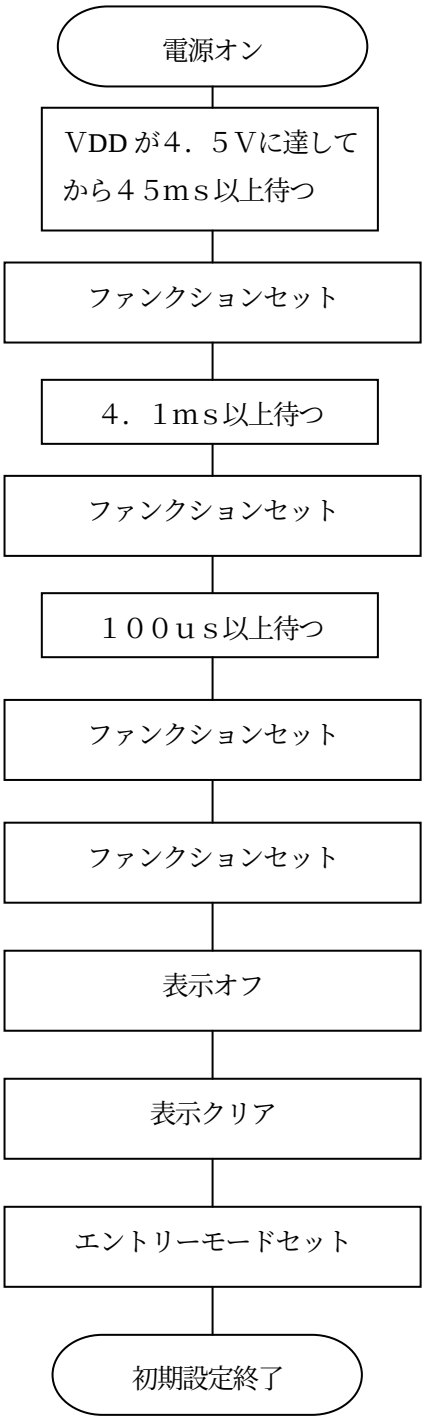
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
１行目	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
２行目	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

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

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宣言です。

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。

１３６～１４１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 “Cat204p4.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  <サンプル>  ポーリング                                                   */
  4: /*                                                                           */
  5: /*  <MOD>       Cat204p4.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-204-KL5C8012 エーワン(株)                                 */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.H"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*      変数宣言                                                             */
 17: /*****************************************************************************/
 18:         Uchar       ModeStep;           /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
 19:         Uchar       Shift;              /* shift ﾊﾟﾀｰﾝ                       */
 20: /*****************************************************************************/
 21: /*      main()                                                               */
 22: /*****************************************************************************/
 23: void    main(void)
 24: {
 25:     outp(SCR1,0xc0);                    /* SYS    ExtMem 0wait ExtIO 1wait  */
 26:     SoftWait1ms(20);                    /* Power On Wait(20ms) 安定待ち     */
 27:     MemInitial();                       /* ﾒﾓﾘ系初期化                       */
 28:     IoInitial();                        /* I/O 系初期化                      */
 29:     while(1) {
 30:         SigInput();                     /* Signal Input  Process             */
 31:         SoftWait1ms(20);                /*       ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り      */
 32:         ModeCntrol();                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                       */
 33:         AllLcdDisp();                   /* LCD    全画面表示                 */
```

```
34:         SigOutput();                    /* Signal Output Process(LED 点灯)   */
35:     }
36: }
37: /*************************************************************************/
38: /*      Ｍｅｍ初期化                                                     */
39: /*************************************************************************/
40: void    MemInitial(void)
41: {
42:     ModeStep = 0;                       /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
43:     Shift = 0;                          /* Led Disp Patan Initial          */
44:
45:     PioMemInitial();                    /* PIO    Mem 初期化               */
46:     LcdMemInitial();                    /* LCD    Mem 初期化               */
47: }
48: /*************************************************************************/
49: /*      Ｉ／Ｏ初期化                                                     */
50: /*************************************************************************/
51: void    IoInitial(void)
52: {
53:     PioIoInitial();                     /* PIO    I/O 初期化               */
54:     LcdIoInitial();                     /* LCD    I/O 初期化               */
55: }
56: /*************************************************************************/
57: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                       */
58: /*************************************************************************/
59: void    ModeCntrol()
60: {
61:     if (GetUpPort(1) & 0x1) {           /* PB[P30] ON?(立上)               */
62:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST    */
63:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ     */
64:     }
65:     switch(ModeStep) {
66:     case 0:
67:         GotoxyMemSet(0,0,"CAT204&BF3000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ      */
68:         GotoxyMemSet(0,1,"  Polling  [P30]");
69:         ModeStep++;
```

```
70:         break;
71:     case 1:
72:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ        */
73:         break;
74:     case 10:                            /*  PIO   TEST                        */
75:         GotoxyMemSet(0,0,"PIO             ");
76:         GotoxyMemSet(0,1,"SW[P40]->SW[P47]");
77:         ModeStep++;
78:         break;
79:     case 11:
80:         PioDemo();
81:         break;
82:     }
83: }
84:
85: /*************************************************************************/
86: /*      RunRun()        CPU 走行表示                                     */
87: /*************************************************************************/
88: void    RunRun()
89: {
90:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
91:     PutOutPort(Shift,'=');
92: }
93: /*************************************************************************/
94: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                 */
95: /*************************************************************************/
96: void    SoftWait1ms(Ushort ms)
97: {
98:     while(ms-- != 0) {
99:         Wait1ms();
100:    }
101: }
102: /*************************************************************************/
103: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                */
104: /*************************************************************************/
105: void    Wait1ms()
```

```
106: {
107:     _asm_("\n          PUSH    HL              \n");
108:     _asm_("\n          LD      HL,1228         \n");   /* 1228*6=7372cyc   */
109:     _asm_("\n W01:                             \n");
110:     _asm_("\n          DEC     HL              \n");   /* cyc = 1          */
111:     _asm_("\n          LD      A,L             \n");   /*     = 1          */
112:     _asm_("\n          OR      H               \n");   /*     = 1          */
113:     _asm_("\n          JP      NZ,W01          \n");   /*     = 3          */
114:     _asm_("\n          POP     HL              \n");   /*    += 6          */
115: }
116: /*************************************************************************/
117: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                */
118: /*************************************************************************/
119: void    SoftWait10us(Ushort us)
120: {
121:     while(us-- != 0) {
122:         Wait10us();
123:     }
124: }
125: /*************************************************************************/
126: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait               */
127: /*************************************************************************/
128: void    Wait10us()
129: {
130:     _asm_("\n          PUSH    HL              \n");
131:     _asm_("\n          LD      HL,13           \n");   /* 13*6=78cyc       */
132:     _asm_("\n W02:                             \n");
133:     _asm_("\n          DEC     HL              \n");   /* cyc = 1          */
134:     _asm_("\n          LD      A,L             \n");   /*     = 1          */
135:     _asm_("\n          OR      H               \n");   /*     = 1          */
136:     _asm_("\n          JP      NZ,W02          \n");   /*     = 3          */
137:     _asm_("\n          POP     HL              \n");   /*    += 6          */
138: }
```

［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。

３３行：

ＬＣＤ全画面表示関数を呼んでいます。

４６行：

“Ｐ＿Ｌｃｄ．ｃ”で使用する変数を初期化する関数を呼んでいます。

５４行：

ＬＣＤの初期設定をする関数を呼んでいます。

６７～６８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。

７５～７６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付いているブザーを利用したいと思います。
評価ボードの図面を見ますと、ＣＮ２－５Ｂ（Ｐ３６／ＯＵＴＢＰ０）にブザーが接続されています。
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、ＡＢＣｗｉｎ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、“Ｃａｔ２０４ｐ５．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？

仕様は、前章に追加したかたちになります。ＰＢ［Ｐ３０］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［Ｐ３０］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［Ｐ３３］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［Ｐ３３］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［Ｐ３２］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［Ｐ３０］　モード開始／終了
ＰＢ［Ｐ３３］　周波数を１００Ｈｚ上げます（ＭＡＸ　１１００Ｈｚ）
ＰＢ［Ｐ３２］　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、ＣＮ２－５Ｂ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、６ファイルの構成になっています。

“ＳｔａｒｔｕｐＢ．ａｓｍ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｐｉｏ２．ｃ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｌｃｄ．ｃ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｔｉｍｅ．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。
“Ｃａｔ２０４ｐ５．ｃ”	メインコントロール部です。

１）Ｔｉｍｅｒコントロール関係

file “P_Time.c”

```
  1: /*************************************************************************/
  2: /*                                                                       */
  3: /*  ＜サンプル＞  ポーリング                                             */
  4: /*                                                                       */
  5: /*  ＜MOD＞       P_Time.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-204-KL5C8012 エーワン(株)                          */
 10: /*                                                                       */
 11: /*************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.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       1000            /*          Max 1000Hz             */
 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:     outp(TCWDA0,0);                     /* TIMA TM0 未使用              */
37:     outp(TCWDA1,0);                     /*      TM1 未使用              */
38:     outp(TCWDB0,0);                     /* TIMB TM0 使用   Buzzer       */
39:     outp(TCWDB1,0);                     /*      TM1 未使用              */
40:     outp(TCWDB2,0);                     /*      TM2 未使用              */
41: }
42: /*************************************************************************/
43: /*      Buzzer   Timer B ﾁｬﾝﾈﾙ 0  OUTBP0 出力に Buzzer 接続               */
44: /*************************************************************************/
45: void    Buzzer(Ushort hz)
46: {
47:     Uchar   cyc;
48:
49:     if ((hz >= BZMIN) && (hz <= BZMAX)) {   /* Buzzer ON                */
50:         outp(TCWDB0,0x1c);              /* outH+PWM+sys/256=28800Hz     */
51:         cyc = 28800 / hz;               /* 周期の計算                   */
52:         outp(TCNTB0,cyc-1);             /* 周期設定                     */
53:         outp(TCNTB0,(cyc/2)-1);         /* 巾設定                       */
54:     }
55:     else {                              /* Buzzer OFF                   */
56:         outp(TCWDB0,0);
57:         outp(TCNTB0,0);
58:         outp(TCNTB0,0);
```

```
59:     }
60: }
61: /*****************************************************************************/
62: /*      TimerDemo   Timer デモ                                               */
63: /*****************************************************************************/
64: void    TimerDemo()
65: {
66:     Uchar   port;
67:     Uchar   dec[4+1];
68:
69:     port = GetUpPort(1);                /* PB[P30]->PB[P33]                  */
70:     if (port & 0xc) {                   /* PB[P32] | PB[P33] ON ?            */
71:         if (port & 0x4) {               /* PB[-P32] ON-立上り                 */
72:             BuzzerHz -= 100;
73:         }
74:         else if (port & 0x8) {          /* PB[+P33] ON-立上り                 */
75:             BuzzerHz += 100;
76:         }
77:         if (BuzzerHz < BZMIN) BuzzerHz = BZMIN;
78:         if (BuzzerHz > BZMAX) BuzzerHz = BZMAX;
79:         Buzzer(BuzzerHz);
80:         Bin2AdecN(dec,BuzzerHz,4);      /* 表示用データ作成                  */
81:         GotoxyMemSet(0,1,dec);
82:     }
83: }
```

リストの説明に入る前に、タイマ関係資料を添付します。

アドレス	ブロック名	ライト時	リード時	シンボル
２８Ｈ	タイマ／カウンタＡ	チャネル０カウンタ	チャネル０カウンタ	ＴＣＮＴＡ０
２９Ｈ		チャネル０コントロールワード	チャネル０ステータス	ＴＣＷＤＡ０
２ＡＨ		チャネル１カウンタ	チャネル１カウンタ	ＴＣＮＴＡ１
２ＢＨ		チャネル１コントロールワード	チャネル１ステータス	ＴＣＷＤＡ１

図［５－２－１］　タイマＡのＩ／Ｏマップ

チャネルｎコントロールワード（ＴＣＷＤＡｎ）		
ﾋﾞｯﾄ	名称	機能
7	－	Ｘ
6	－	Ｘ
5		０
4		０
3		０
2	ＰＵＬＳＥ	１：パルス出力 ０：トグル出力
1	－	Ｘ
0	ＣＯＵＮＴＣＬＫ	１：システムクロック（ＣＬＫ） ０：外部クロック（ＸＣＬＫ）

図［５－２－２］　タイマＡの分周モード　ｎ＝０，１

チャネルｎコントロールワード（ＴＣＷＤＡｎ）		
ﾋﾞｯﾄ	名称	機能
7	－	Ｘ
6	－	Ｘ
5		０
4		０
3		１
2 1	ＰＥＲＩＯＤ	１１：（2^{12}＋１）カウンタクロック １０：（2^{10}＋１）カウンタクロック ０１：（2^{8}＋１）カウンタクロック ００：（2^{6}＋１）カウンタクロック
0	ＣＯＵＮＴＣＬＫ	１：システムクロック（ＣＬＫ） ０：外部クロック（ＸＣＬＫ）

図［５－２－３］　タイマＡのパルス幅変調（ＰＷＭ）モード　ｎ＝０，１

チャネルnコントロールワード（ＴＣＷＤＡｎ）		
ﾋﾞｯﾄ	名称	機能
7	－	Ｘ
6	－	Ｘ
5		０
4		１
3	ＳＴＲＯＢＥ	１：ストローブ出力 ０：ワンショット出力
2	ＴＲＩＧＧＥＲ	１：ハード・トリガ ０：ソフト・トリガ
1	ＲＥＶＥＲＳＥ	１：出力反転する ０：出力反転しない
0	ＣＯＵＮＴＣＬＫ	１：システムクロック（ＣＬＫ） ０：外部クロック（ＸＣＬＫ）

図［５－２－４］　タイマAのパルスモード　ｎ＝０，１

チャネルnコントロールワード（ＴＣＷＤＡｎ）		
ﾋﾞｯﾄ	名称	機能
7	－	Ｘ
6	－	Ｘ
5		１
4		０
3	ＲＥＰＥＡＴ	１：連続測定 ０：一回測定
2	ＣＬＥＡＲ	カウンタを０ｘｆｆｆｆにクリア １：ＧＡＴＥ信号の立上りエッジ ０：ＧＡＴＥ信号の立下りエッジ
1	ＦＯＲＷＡＲＤ	カウンター＞ＣＲレジスタへの転送 １：ＧＡＴＥ信号の立上りエッジ ０：ＧＡＴＥ信号の立下りエッジ
0	ＣＯＵＮＴＣＬＫ	１：システムクロック（ＣＬＫ） ０：外部クロック（ＸＣＬＫ）

図［５－２－５］　タイマAのパルス幅／周波測定モード　ｎ＝０，１

アドレス	ブロック名	ライト時	リード時	シンボル
２０H	タイマ／カウンタB	チャネル０カウンタ	チャネル０カウンタ	ＴＣＮＴＢ０
２１H		チャネル０コントロールワード	チャネル０ステータス	ＴＣＷＤＢ０
２２H		チャネル１カウンタ	チャネル１カウンタ	ＴＣＮＴＢ１
２３H		チャネル１コントロールワード	チャネル１ステータス	ＴＣＷＤＢ１
２４H		チャネル２カウンタ	チャネル２カウンタ	ＴＣＮＴＢ２
２５H		チャネル２コントロールワード	チャネル２ステータス	ＴＣＷＤＢ２

図［5－2－6］　タイマBのＩ／Ｏマップ

チャネルnコントロールワード（ＴＣＷＤＢn）		
ﾋﾞｯﾄ	名称	機能
7	－	Ｘ
6	－	Ｘ
5	－	Ｘ
4	TOGGLE	１：ＯＵＴＰ端子の初期出力を“Ｈ”にする ０：ＯＵＴＰ端子の初期出力を“Ｌ”にする
3 2	COUNTMODE	１１：PWMモード １０：WDTモード ０１：連続カウントモード ００：単発カウントモード
1 0	COUNTCLK	プリスケーラのシステムクロックに対しての分周レート設定 １１：４分周（ＧＡＴＥ機能有） １０：４分周（ＧＡＴＥ機能無） ０１：１６分周（ＧＡＴＥ機能無） ００：２５６分周（ＧＡＴＥ機能無）

図［5－2－7］　タイマBのモード設定　ｎ＝０，１，２

［リストの説明］

１８行：

出力周波数の最小値の定義です。

１９行：

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。

３４～４１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。

ただし、ここでは全タイマー未動作状態で初期化します。

ここのサンプルでは、タイマＢのチャネル０のみ使用します。

４５～６０行：

指定周波数のパルスを出力する関数です。

指定周波数が、範囲外だった場合、パルス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、カウンタ／モード設定します。

タイマＢのチャネル０をモード設定します。図［５－２－４］参照

分周レート：２５６　PWMモード　ＯＵＴＰ端子の初期を“Ｈ”にする

［周期計算式］

（システムクロック÷分周レート）÷指定周波数＝周期

周期÷２＝周期巾（デューティ５０％）

システムクロック＝７．３７２８MHｚ

７．３７２８MHｚ÷２５６＝２８８００Hｚ

２８８００÷指定周波数＝周期

になります。

６４～８３行：

ＰＢ［Ｐ３２］，ＰＢ［Ｐ３３］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: /*  ＜使用ﾊｰﾄﾞ＞ CAT-204-KL5C8012 エーワン(株)                            */
 10: /*                                                                       */
 11: /*************************************************************************/
 12: #include    <CTYPE.H>
 13: #include    "CAT204.H"
 14: #include    "DemoCtl.h"
 15: /*************************************************************************/
 16: /*       ＢＩＮ－＞アスキーＤＥＣ変換 桁指定  asc[keta]=NULL 付き        */
 17: /*************************************************************************/
 18: void    Bin2AdecN(char *asc,Ushort bin,Uchar keta)
 19: {
 20:     asc[keta] = 0;
 21:     while(keta--) {
 22:         asc[keta]  = bin % 10;
 23:         bin /= 10;
 24:         asc[keta] |= '0';
 25:     }
 26: }
 27: /*************************************************************************/
 28: /*      ＢＩＮ－＞アスキーＨＥＸ変換 桁指定  asc[keta]=NULL 付き        */
 29: /*************************************************************************/
 30: void    Bin2AhexN(char *asc,Ushort bin,Uchar keta)
 31: {
 32:     Uchar  dt;
 33:
```

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

```
70:         else            dt = c - 0x30;
71:         *bin = (*bin << 4) | dt;
72:         ptr++;
73:     }
74:     return(ptr);
75: }
76: /*************************************************************************/
77: /*      ストリングＣｏｐｙ(WORD)                                          */
78: /*************************************************************************/
79: Ushort * _strcpyW(Ushort *dst,Ushort *src)
80: {
81:     while(*src != 0) {
82:         *dst++ = *src++;
83:     }
84:     *dst = 0;
85:     return(dst);
86: }
```

［リストの説明］

１８～２６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。

３０～４２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。

４６～５７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。

６１～７５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。

７９～８６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 “Cat204p5.c”

```
/*****************************************************************************/
/*                                                                           */
/*  <サンプル>  ポーリング                                                   */
/*                                                                           */
/*  <MOD>       Cat204p5.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-204-KL5C8012 エーワン(株)                                 */
/*                                                                           */
/*****************************************************************************/
#include    <machine.H>
#include    "CAT204.H"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      外部変数使用                                                         */
/*****************************************************************************/
extern  short       BuzzerHz;           /* TIMER  Buzzer Hz                  */
/*****************************************************************************/
/*      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)
{
    outp(SCR1,0xc0);                    /* SYS    ExtMem 0wait ExtIO 1wait   */
    outp(SCR0,0x10);                    /*        (OUTBx,OUTAx)              */
    SoftWait1ms(20);                    /* Power On Wait(20ms) 安定待ち      */
    MemInitial();                       /* ﾒﾓﾘ系初期化                       */
    IoInitial();                        /* I/O 系初期化                      */
```

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

```
70:         else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST     */
71:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ       */
72:         Buzzer(0);                                  /*  強制 OFF            */
73:     }
74:     switch(ModeStep) {
75:     case 0:
76:         GotoxyMemSet(0,0,"CAT204&BF3000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ       */
77:         GotoxyMemSet(0,1,"Polling    [P30]");
78:         ModeStep++;
79:         break;
80:     case 1:
81:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ       */
82:         break;
83:     case 10:                            /*  PIO    TEST                     */
84:         GotoxyMemSet(0,0,"PIO             ");
85:         GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
86:         ModeStep++;
87:         break;
88:     case 11:
89:         PioDemo();
90:         break;
91:     case 20:                            /*  Timer TEST                  */
92:         GotoxyMemSet(0,0,"Timer/Counter   ");
93:         GotoxyMemSet(0,1,"0000Hz[+P33-P32]");
94:         BuzzerHz = 0;                   /* Buzzer Hz            */
95:         ModeStep++;
96:         break;
97:     case 21:
98:         TimerDemo();
99:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ       */
100:        break;
101:    }
102: }
103: /*************************************************************************/
104: /*      RunRun()        CPU 走行表示                                     */
105: /*************************************************************************/
```

```
106: void    RunRun()
107: {
108:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
109:     PutOutPort(Shift,'=');
110: }
111: /*************************************************************************/
112: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                  */
113: /*************************************************************************/
114: void    SoftWait1ms(Ushort ms)
115: {
116:     while(ms-- != 0) {
117:         Wait1ms();
118:     }
119: }
120: /*************************************************************************/
121: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                */
122: /*************************************************************************/
123: void    Wait1ms()
124: {
125:     _asm_("\n           PUSH    HL              \n");
126:     _asm_("\n           LD      HL,1228         \n");   /* 1228*6=7372cyc   */
127:     _asm_("\n W01:                              \n");
128:     _asm_("\n           DEC     HL              \n");   /* cyc = 1          */
129:     _asm_("\n           LD      A,L             \n");   /*     = 1          */
130:     _asm_("\n           OR      H               \n");   /*     = 1          */
131:     _asm_("\n           JP      NZ,W01          \n");   /*     = 3          */
132:     _asm_("\n           POP     HL              \n");   /*    += 6          */
133: }
134: /*************************************************************************/
135: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                */
136: /*************************************************************************/
137: void    SoftWait10us(Ushort us)
138: {
139:     while(us-- != 0) {
140:         Wait10us();
141:     }
```

```
142: }
143: /*************************************************************************/
144: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                */
145: /*************************************************************************/
146: void    Wait10us()
147: {
148:     _asm_("\n           PUSH    HL              \n");
149:     _asm_("\n           LD      HL,13           \n");   /* 13*6=78cyc       */
150:     _asm_("\n W02:                              \n");
151:     _asm_("\n           DEC     HL              \n");   /* cyc = 1          */
152:     _asm_("\n           LD      A,L             \n");   /*     = 1          */
153:     _asm_("\n           OR      H               \n");   /*     = 1          */
154:     _asm_("\n           JP      NZ,W02          \n");   /*     = 3          */
155:     _asm_("\n           POP     HL              \n");   /*    += 6          */
156: }
```

［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。

タイマーを使用することになりましたので、システムコントロールレジスタの設定が必要になりました。

１８行：

このモジュールで使用する外部変数宣言です。

３０行：

システムコントロールレジスタ（ＳＣＲ０）のＤ４（端子９７～１００の機能）を
ＯＵＴＢＳ２，ＯＵＴＢＰ０，ＯＵＴＡ１，ＯＵＴＡ０として機能する側にする。
図［５－２－８］を参照

５２行：

“Ｐ＿Ｔｉｍｅ．ｃ”で使用する変数の初期化関数を呼んでいます。

６１行：

タイマレジスタを初期化する関数を呼んでいます。

９１～１００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。

システムコントロールレジスタ（ＳＣＲ０）		
ﾋﾞｯﾄ	名称	機能
7	タイマAの動作	１：タイマAのチャネル１，０がカスケード接続され、３２ビットカウンタとして動作します ０：タイマAのチャネル１，０が独立に動作します
6	タイマAチャネル１ ＧＡＴＥ入力 （ＧＡＴＥＡ１）	１：端子７６から入力されます ０：常に“Ｈ”が入力されます。端子７６はＰ１２として機能する
5	タイマAチャネル０ ＧＡＴＥ入力 （ＧＡＴＥＡ０）	１：端子７９から入力されます ０：常に“Ｈ”が入力されます。端子７９はＰ１０として機能する
4	端子９７～１００機能	１：ＯＵＴＢＳ２，ＯＵＴＢＰ０，ＯＵＴＡ１，ＯＵＴＡ０として機能します ０：Ｐ３４～Ｐ３７として機能します
3	端子1，2，4，5，6の機能	１：ＯＵＴＢＳ０，ＯＵＴＢＰ１，ＯＵＴＢＰ２，ＳＹＮＣとして機能します ０：Ｐ３０～Ｐ３３として機能します
2	端子８９，９３の機能	１：ＳＹＮＤＢＤ，ＳＹＤＴＩＮとして機能します。 ０：Ｐ４３，Ｐ４７として機能します。 ＳＹＤＴＩＮには“Ｌ”が入力されます
1	端子９０，９１，９４，９５の機能	１：ＤＳＲ＿，ＣＴＳ＿，ＤＴＲ＿，ＲＴＳ＿として機能します ０：Ｐ４１，Ｐ４２，Ｐ４５，Ｐ４６として機能します ＤＳＲ＿，ＣＴＳ＿には“Ｌ”が入力されます
0	シリアルポート ＴＸＣ，ＲＸＣ入力	１：端子７３，７４からの入力を使います ０：タイマＢチャネル１のＯＵＴＢＰ出力を使用します 端子７３，７４はＰ１４，Ｐ１５として機能します

図［5－2－8］　システムコントロールレジスタ０

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、わかって頂けたでしょうか？（Ｐ＿Ｔｉｍｅ．ｃのどこか）

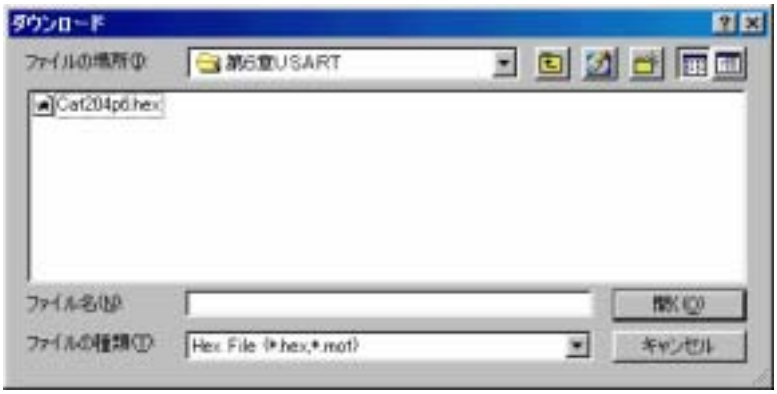
興味のあるかたは、修正して“ｋｍｍａｋｅ”を実行してチャレンジしてみて下さい。

次は、ＵＳＡＲＴ使用例の解説へと進みます。

第6章　ＵＳＡＲＴ

ここの章では、ＵＳＡＲＴのイニシャルとＵＳＡＲＴ使用サンプルの解説をします。
ＵＳＡＲＴ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。

まずは、ＡＢＣｗｉｎ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ＵＳＡＲＴ¥
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、**“Ｃａｔ２０４ｐ６．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“Ｕｓａｒｔ”と表示がでるまで、**
ＰＢ［Ｐ３０］を押して下さい。
ここで、評価ボード上の**ＳＷ１１－１をオン**（ＴＸＤ，ＲＸＤを折り返す）にして下さい。
ＰＢ［Ｐ３１］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［Ｐ３１］停止を押されるまで、送受信を繰り返します。

ＬＣＤ画面　　ＵｓａｒｔＴｘｘＲｘｘ［Ｐ３１］
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［Ｐ３２］、ＰＢ［Ｐ３３］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、７ファイルの構成になっています。

“ＳｔａｒｔｕｐＢ．ａｓｍ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｐｉｏ２．ｃ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｌｃｄ．ｃ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｔｉｍｅ．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。
“Ｐ＿Ｕｓａｒｔ．ｃ”	ＵＳＡＲＴコントロールのモジュールです。
“Ｃａｔ２０４ｐ５．ｃ”	メインコントロール部です。

１）ＵＳＡＲＴコントロール関係

file “P_Usart.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞  ポーリング                                                 */
  4: /*                                                                           */
  5: /*  ＜MOD＞       P_Usart.c                                                  */
  6: /*  ＜役割＞      USART(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-KL5C8012 エーワン(株)                              */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.H"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*      USART ボレート周期計算+ETC                                           */
 17: /*                    Sys      = 7372800Hz                                   */
 18: /*                    TIMERB   = 4 分周                                      */
 19: /*                    TXC/RXC = 1/16                                         */
 20: /*****************************************************************************/
 21: #define CLOCK       (7372800/4)         /* 調歩同期 ﾎﾞｰﾚｰﾄｼﾞｪﾈﾚｰﾄ計算       */
```

```
22: #define BRRN(b)     (CLOCK/(b*16))
23: #define NON         0                   /* ﾊﾟﾘﾃｨ     NON                 */
24: #define EVEN        1                   /*           EVEN                */
25: #define ODD         2                   /*           ODD                 */
26: /*****************************************************************************/
27: /*      変数宣言                                                             */
28: /*****************************************************************************/
29:         Uchar       K51Step;            /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                    */
30:         Uchar       K51Select;          /* ﾎﾞｰﾚｰﾄ選択                      */
31:         Uchar       TxDat;              /* 送信ﾃﾞｰﾀ                        */
32:         Uchar       RxDat;              /* 受信ﾃﾞｰﾀ                        */
33: const   Ushort      BpsTbl[] =
34: {
35:         2400,
36:         4800,
37:         9600,
38:         19200,
39:         38400,
40:         0,
41: };
42: /*****************************************************************************/
43: /*      Ｍｅｍ初期化                                                         */
44: /*****************************************************************************/
45: void    K51MemInitial(void)
46: {
47:     K51Step   = 0;                      /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                    */
48:     K51Select = 2;                      /* 9600BPS                         */
49: }
50: /*****************************************************************************/
51: /*      Ｉ／Ｏ初期化                                                         */
52: /*****************************************************************************/
53: void    K51IoInitial(void)
54: {
55:     /* 重要!! SCR0 の"D0"を"0"にする */
56:     Rs0pen(9600,8,NON);
57: }
```

```
58: /*************************************************************************/
59: /*      RS232C  Open  bps    = ﾎﾞﾚｰﾄ[2400,4800,*9600,19200,38400]          */
60: /*                    ch     = ｷｬﾗｸﾀ[7,*8]                                 */
61: /*                    pty    = ﾊﾟﾘﾃｨ[*0=NON 1=EVEN 2=ODD]                  */
62: /*                    *      = ﾃﾞﾌｫﾙﾄ                                      */
63: /*                    固定   = x16 STOP=1bit                               */
64: /*                    ﾎﾞｰﾚｰﾄ = ﾀｲﾏ B ﾁｬﾈﾙ 1 の OUTBP を使用                  */
65: /*************************************************************************/
66: void    RsOpen(Ushort bps,Uchar ch,Uchar pty)
67: {
68:     Uchar   cyc;
69:     Uchar   mode;
70:
71:     cyc = BRRN(9600);                   /* ﾎﾞｰﾚｰﾄ ｼﾞｪﾈﾚｰﾄ計算 (ﾃﾞﾌｫﾙﾄｾｯﾄ)   */
72:     if (bps == 2400)        cyc = BRRN(2400);
73:     else if (bps == 4800)   cyc = BRRN(4800);
74:     else if (bps == 9600)   cyc = BRRN(9600);
75:     else if (bps == 19200)  cyc = BRRN(19200);
76:     else if (bps == 38400)  cyc = BRRN(38400);
77:
78:     outp(TCWDB1,0x0e);                  /* TIME-B1 outL+PWM+sys/4          */
79:     outp(TCNTB1,cyc-1);                 /*         周期設定                */
80:     outp(TCNTB1,(cyc/2)-1);             /*         巾設定(ﾃﾞｭｰﾃｨ 50%)       */
81:
82:     outp(K51COM0,0);                    /* KP51  ﾀﾞﾐｰ                      */
83:     outp(K51COM0,0);                    /*       ﾀﾞﾐｰ                      */
84:     outp(K51COM0,0);                    /*       ﾀﾞﾐｰ                      */
85:     outp(K51COM0,0x40);                 /*       ｿﾌﾄﾘｾｯﾄ                   */
86:     mode = 0x4e;                        /* Mode  Stop=1 NON 8bit x16(ﾃﾞﾌｫﾙﾄ)*/
87:     if (pty == EVEN)      mode |= 0x30;     /*  ﾊﾟﾘﾃｨ=EVEN                */
88:     else if (pty == ODD)  mode |= 0x20;     /*  ﾊﾟﾘﾃｨ=ODD                 */
89:     if (ch == 7)          mode &= ~(0x4);   /*  ｷｬﾗｸﾀ 7                   */
90:     outp(K51COM0,mode);                 /*       設定                      */
91:     outp(K51COM0,0x37);                 /* Cmd   RTS ERR RX=EI DTR TX=EI   */
92: }
93: /*************************************************************************/
```

```
 94: /*      RsPutch     RS232C 送信                                          */
 95: /*****************************************************************************/
 96: void    RsPutch(Uchar tx)
 97: {
 98:     while((inp(K51COM0) & 0x4) == 0) {} /* TXEMPTY 待ち                  */
 99:     outp(K51DAT0,tx);
100: }
101: /*****************************************************************************/
102: /*      RsGetch     RS232C 受信                                          */
103: /*****************************************************************************/
104: short    RsGetch()
105: {
106:     Ushort  time;
107:     Uchar   dt;
108:
109:     time = 0;
110:     while((inp(K51COM0) & 0x3a) == 0) { /* FE+OE+PE+RXRDY   ON ?        */
111:         SoftWait1ms(1);                 /* 1ms                          */
112:         if (++time >= 20) return(-1);   /* Error                        */
113:     }
114:     dt = inp(K51DAT0);
115:     if ((inp(K51COM0) & 0x38) != 0 ) {  /* FE+OE+PE ON ?                */
116:         outp(K51COM0,0x37);             /* ErrReset RTS ERR RX=EI DTR TX=EI */
117:         return(-1);
118:     }
119:     return(dt);
120: }
121: /*****************************************************************************/
122: /*      UsartDemo()     ＵＳＡＲＴデモ                                     */
123: /*****************************************************************************/
124: void    K51Demo()
125: {
126:     Uchar   dec[2+1];
127:     short   stat;
128:
129:     K51Sequence();                      /* Usart 操作ｺﾝﾄﾛｰﾙ              */
```

```
130:     switch(K51Step) {
131:     case 0:
132:         break;
133:     case 1:
134:         RsOpen(BpsTbl[K51Select],8,0);  /* RS232C Open                  */
135:         GotoxyMemSet(5,0,"TxxRxx");
136:         TxDat = 0;
137:         K51Step++;
138:         break;
139:     case 2:
140:         Bin2AhexN(dec,TxDat,2);         /* 表示用送信データ作成          */
141:         GotoxyMemSet(6,0,dec);
142:         RsPutch(TxDat++);               /* 送信                         */
143:         K51Step++;
144:         break;
145:     case 3:
146:         stat = RsGetch();
147:         if (stat == -1) GotoxyMemSet(9,0,"ee");     /* Error            */
148:         else {
149:             RxDat = stat;
150:             Bin2AhexN(dec,RxDat,2);     /* 表示用受信データ作成          */
151:             GotoxyMemSet(9,0,dec);
152:         }
153:         K51Step = 2;
154:         break;
155:     }
156: }
157: /*****************************************************************************/
158: /*      K51Sequence()   Usart 操作ｺﾝﾄﾛｰﾙ                                     */
159: /*****************************************************************************/
160: void    K51Sequence()
161: {
162:     Uchar   port;
163:     Uchar   dec[5+1];
164:
165:     port = GetUpPort(1);                /* PB[P30]->PB[P33]             */
```

```
166:     if (port & 0xe) {                  /* PB[P31]->PB[P33] ON(立上) ?      */
167:         if (port & 0x2) {              /* PB[P31] ON ? 送信ｽﾀｰﾄ/ｽﾄｯﾌﾟ      */
168:             if (K51Step == 0) K51Step = 1;
169:             else              K51Step = 0;
170:         }
171:         if (K51Step == 0) {            /* 停止中のみ受け付ける             */
172:             if (port & 0x4) {          /* PB[P32] ON ? ﾎﾞﾚｰﾄ下げ?          */
173:                 if (K51Select != 0) --K51Select;
174:             }
175:             else if (port & 0x8) {     /* PB[P33] ON ? ﾎﾞﾚｰﾄ上げ?          */
176:                 if (BpsTbl[K51Select+1] != 0) ++K51Select;
177:             }
178:             Bin2AdecN(dec,BpsTbl[K51Select],5); /* 表示用データ作成        */
179:             GotoxyMemSet(0,1,dec);
180:         }
181:     }
182: }
```

リストの説明に入る前に、ＵＳＡＲＴ関係資料を添付します。

アドレス	ブロック名	ライト時	リード時	シンボル
３８Ｈ	ＵＳＡＲＴ	送信データ	受信データ	Ｋ５１ＤＡＴ０
３９Ｈ		モード／コントロール	ステータス	Ｋ５１ＣＯＭ０

図［6－2－1］　ＵＳＡＲＴのＩ／Ｏマップ

モードレジスタ（Ｋ５１ＣＯＭ０）　ＷＲＩＴＥ		
ﾋﾞｯﾄ	名称	機能
7 6	ストップビット	００：無効 ０１：１ビット １０：１．５ビット １１：２ビット
5	パリティビット	１：偶数（ＥＶＥＮ） ０：奇数（ＯＤＤ）
4	パリティイネーブル	１：イネーブル ０：ディセーブル
3 2	キャラクタ長	００：５ビット ０１：６ビット １０：７ビット １１：８ビット
1 0	ボーレート	０１：ｘ１ １０：ｘ１６ １１：ｘ６４

図［6－2－2］　ａｓｙｎｃモード設定

コマンドレジスタ（Ｋ５１ＣＯＭ０）　ＷＲＩＴＥ		
ﾋﾞｯﾄ	名称	機能
7	ＥＨ	１：ハントモード（同期キャラクタの検出）に入る
6	ＳＲＥＳ	１：ソフトウェアリセット（内部初期化）
5	ＲＴＳ	１：ＲＴＳ＿“Ｌ”出力（ＯＮ） ０：ＲＴＳ＿“Ｈ”出力（ＯＦＦ）
4	ＥＲＲ	１：エラーリセット（ＰＥ＋ＯＥ＋ＦＥ）
3	ＳＢＲＫ	１：ブレーク信号送信（ＴＸＤ＝“Ｌ”） ０：通常
2	ＲＸＥＮ	１：受信イネーブル ０：受信ディセーブル
1	ＤＴＲ	１：ＤＴＲ＿は“Ｌ”出力（ＯＮ） ０：ＤＴＲ＿は“Ｈ”出力（ＯＦＦ）
0	ＴＸＥＮ	１：送信イネーブル ０：送信ディセーブル

図［6－2－3］　コマンドレジスタ

ステータスレジスタ（Ｋ５１ＣＯＭ０）　ＲＥＡＤ		
ﾋﾞｯﾄ	名称	機能
7	ＤＳＲ	０：ＤＳＲ＿は“Ｈ”入力（ＯＦＦ） １：ＤＳＲ＿は“Ｌ”入力（ＯＮ）
6	ＳＹＮＣ／ ＢＲＫ	０：通常 １：同期／ブレーク信号検出
5	ＦＥ	０：通常 １：フレミングエラー検出
4	ＯＥ	０：通常 １：オーバーランエラー検出
3	ＰＥ	０：通常 １：パリティエラー検出
2	ＴＸＥＭＰ	０：第１，２バッファに送信データ有り １：第１，２バッファの送信データ空
1	ＲＸＲＤＹ	０：受信データ無し １：受信データ有り（１キャラクタ）
0	ＴＸＲＤＹ	０：送信データバッファ（第２バッファ有）送信データセット不可 １：送信データバッファ（第２バッファ空）送信データセット可

図［６－２－４］　ステータスレジスタ

［リストの説明］

２１～２２行：

ボーレートよりタイマＢ（PWMモード）にあたえるための周期計算のマクロです。

（システムクロック÷分周レート）÷（指定ボーレート×１６）＝周期

タイマBのチャネル１を分周レート　：４

PWMモード　ＯＵＴＰ端子の初期を：Ｌ

システムクロック　　：７．３７２８MHｚ

（７．３７２８MHｚ÷４＝１８４３２００Hｚ）

（指定ボーレート×１６＝出力周波数）

になります。

２９行：

送受信をコントロールする変数宣言です。

３０行：

ボーレートを選択する変数宣言です。

３１行：

送信データを記憶する変数宣言です。

３２行：

受信データを記憶する変数宣言です。

３３～４１行：

選択できるボーレートテーブルの変数宣言です。

４５～４９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。

５３～５７行：

ＵＳＡＲＴを初期化する関数“ＲｓＯｐｅｎ”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。

６６～９２行：

ＵＳＡＲＴを初期化する関数です。

ハードの構成上、シリアルポートに必要な**ＴＸＣ，ＲＸＣの入力をタイマBチャネル１のＯＵＴＢＰ出力を使用**するに選択をする必要があります。

選択方法は、ＳＣＲ０（システムコントロールレジスタ０）のDOを“０”にします。

“Ｃａｔ２０４ｐ６．ｃ”の２６行と図［5－2－8］を参照

ＳＣＲ０への設定は、ｍａｉｎ関数の最初で実行しているため、ここでは設定しません。

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］
　　モードでは、5，6も設定できますがここでは無視します。
第3引数は、パリティ［0＝NON（ディセーブル）　1＝偶数　2＝奇数］

入力周波数（TXC／RXC）に対する分周は、x16固定とします。
ストップビットは、1ビット固定とします。

［71～80行］

タイマBチャネル1のOUTBP出力をTXC，RXCに与えるため、指定ボーレートより周期を計算し（マクロ使用）、タイマBチャネル1をモード設定します。図［5－2－4］参照
タイマBのチャネル1を分周レート　：4
PWMモード　OUTP端子の初期を：L
周期÷2＝周期巾（デューティ50%）

あとは、求めた周期を、タイマBのチャネル1の内部レジスタに設定します。

［82～91行］

USART（KP51）を初期化します。
どのような状態でも、USARTのモード設定ができるように、“0”を3回セットしてから、ソフトウェアリセットコマンドを発行しています。
あとは、指定引数にもとずきモード・コマンドの順に設定します。

96～100行：

1バイトデータを送信する関数です。

104～120行：

1バイトデータを受信する関数です。
RXRDY（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。

124～156行：

USARTの動作確認をするための関数です。
USART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。

160～182行：

USART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 “Cat204p6.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞ ポーリング                                                  */
  4: /*                                                                           */
  5: /*  ＜MOD＞      Cat204p6.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-204-KL5C8012 エーワン(株)                               */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.H"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*      外部変数使用                                                         */
 17: /*****************************************************************************/
 18: extern  short      BuzzerHz;           /* TIMER  Buzzer Hz                 */
 19: extern  Uchar      K51Step;            /* USART  ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ               */
 20: extern  Uchar      K51Select;          /*        ﾎﾞｰﾚｰﾄ選択                */
 21: /*****************************************************************************/
 22: /*      変数宣言                                                             */
 23: /*****************************************************************************/
 24:         Uchar      ModeStep;          /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
 25:         Uchar      Shift;             /* shift ﾊﾟﾀｰﾝ                       */
 26: /*****************************************************************************/
 27: /*      main()                                                               */
 28: /*****************************************************************************/
 29: void    main(void)
 30: {
 31:     outp(SCR1,0xc0);                  /* SYS    ExtMem 0wait ExtIO 1wait  */
 32:     outp(SCR0,0x10);                  /*        (OUTBx,OUTAx)             */
 33:     SoftWait1ms(20);                  /* Power On Wait(20ms) 安定待ち     */
```

```
34:     MemInitial();                   /* ﾒﾓﾘ系初期化                      */
35:     IoInitial();                    /* I/O 系初期化                     */
36:     while(1) {
37:         SigInput();                 /* Signal Input  Process            */
38:         SoftWait1ms(20);            /*       ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り     */
39:         ModeCntrol();               /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                       */
40:         AllLcdDisp();               /* LCD    全画面表示                */
41:         SigOutput();                /* Signal Output Process(LED 点灯)  */
42:     }
43: }
44: /*************************************************************************/
45: /*      Ｍｅｍ初期化                                                     */
46: /*************************************************************************/
47: void    MemInitial(void)
48: {
49:     ModeStep = 0;                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
50:     Shift = 0;                      /* Led Disp Patan Initial           */
51:
52:     PioMemInitial();                /* PIO    Mem 初期化                */
53:     LcdMemInitial();                /* LCD    Mem 初期化                */
54:     TimMemInitial();                /* TIM    Mem 初期化                */
55:     K51MemInitial();                /* SIO    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:     K51IoInitial();                 /* SIO    I/O 初期化                */
66: }
67: /*************************************************************************/
68: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                       */
69: /*************************************************************************/
```

```
 70: void    ModeCntrol()
 71: {
 72:     if (GetUpPort(1) & 0x1) {           /* PB[P30] ON?(立上)                */
 73:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST     */
 74:         else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST     */
 75:         else if (ModeStep < 30) ModeStep = 30;      /*  USART Goto TEST     */
 76:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ      */
 77:         Buzzer(0);                                  /*  強制 OFF            */
 78:     }
 79:     switch(ModeStep) {
 80:     case 0:
 81:         GotoxyMemSet(0,0,"CAT204&BF3000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ       */
 82:         GotoxyMemSet(0,1,"Polling    [P30]");
 83:         ModeStep++;
 84:         break;
 85:     case 1:
 86:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ      */
 87:         break;
 88:     case 10:                            /*  PIO   TEST                      */
 89:         GotoxyMemSet(0,0,"PIO             ");
 90:         GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
 91:         ModeStep++;
 92:         break;
 93:     case 11:
 94:         PioDemo();
 95:         break;
 96:     case 20:                            /*  Timer TEST                      */
 97:         GotoxyMemSet(0,0,"Timer/Counter   ");
 98:         GotoxyMemSet(0,1,"0000Hz[+P33-P32]");
 99:         BuzzerHz = 0;                               /* Buzzer Hz           */
100:         ModeStep++;
101:         break;
102:     case 21:
103:         TimerDemo();
104:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ      */
105:         break;
```

```
106:     case 30:                              /*  USART TEST                      */
107:         GotoxyMemSet(0,0,"Usart 8,n,1[P31]");
108:         GotoxyMemSet(0,1,"09600b[+P33-P32]");
109:         K51Step   = 0;                                  /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ          */
110:         K51Select = 2;                                  /* 9600BPS             */
111:         ModeStep++;
112:         break;
113:     case 31:
114:         K51Demo();
115:         RunRun();                         /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ       */
116:         break;
117:     }
118: }
119:
120: /*************************************************************************/
121: /*      RunRun()        CPU 走行表示                                     */
122: /*************************************************************************/
123: void    RunRun()
124: {
125:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
126:     PutOutPort(Shift,'=');
127: }
128: /*************************************************************************/
129: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                 */
130: /*************************************************************************/
131: void    SoftWait1ms(Ushort ms)
132: {
133:     while(ms-- != 0) {
134:         Wait1ms();
135:     }
136: }
137: /*************************************************************************/
138: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                */
139: /*************************************************************************/
140: void    Wait1ms()
141: {
```

```
142:     _asm_("\n          PUSH    HL              \n");
143:     _asm_("\n          LD      HL,1228         \n");   /* 1228*6=7372cyc   */
144:     _asm_("\n W01:                             \n");
145:     _asm_("\n          DEC     HL              \n");   /* cyc = 1          */
146:     _asm_("\n          LD      A,L             \n");   /*     = 1          */
147:     _asm_("\n          OR      H               \n");   /*     = 1          */
148:     _asm_("\n          JP      NZ,W01          \n");   /*     = 3          */
149:     _asm_("\n          POP     HL              \n");   /*    += 6          */
150: }
151: /*****************************************************************************/
152: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                    */
153: /*****************************************************************************/
154: void    SoftWait10us(Ushort us)
155: {
156:     while(us-- != 0) {
157:         Wait10us();
158:     }
159: }
160: /*****************************************************************************/
161: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                   */
162: /*****************************************************************************/
163: void    Wait10us()
164: {
165:     _asm_("\n          PUSH    HL              \n");
166:     _asm_("\n          LD      HL,13           \n");   /* 13*6=78cyc       */
167:     _asm_("\n W02:                             \n");
168:     _asm_("\n          DEC     HL              \n");   /* cyc = 1          */
169:     _asm_("\n          LD      A,L             \n");   /*     = 1          */
170:     _asm_("\n          OR      H               \n");   /*     = 1          */
171:     _asm_("\n          JP      NZ,W02          \n");   /*     = 3          */
172:     _asm_("\n          POP     HL              \n");   /*    += 6          */
173: }
```

［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。

１８～２０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。

５５行：

“Ｐ＿Ｕｓａｒｔ．ｃ”で使用する変数の初期化関数を呼んでいます。

６５行：

ＵＳＡＲＴを初期化する関数を呼んでいます。

１０６～１１６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、ＡＢＣｗｉｎ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、**“Ｃａｔ２０４ｐ．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［Ｐ３０］を押して下さい。
ＬＣＤの下行　Ｐ３１［Ｍ］３３［ス］３２［セ］　と表示しているはずです。
Ｐ３１［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［Ｐ３１］を押して下さい。（モード変更）
Ｐ３１［Ａ］と表示したはずです。（Ａｕｔｏ演奏の意味）

ＰＢ［Ｐ３２］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［Ｐ３３］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。

“ＳｔａｒｔｕｐＢ．ａｓｍ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｐｉｏ２．ｃ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｌｃｄ．ｃ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｔｉｍｅ．ｃ”	前章のまま使用したので解説を省略します。
“Ｐ＿Ｕｓａｒｔ．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。
“Ｐ＿Ｍｅｌｏｄｙ．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 “P_Melody.c”

```
/*****************************************************************************/
/*                                                                           */
/*  <サンプル>  ポーリング                                                   */
/*                                                                           */
/*  <MOD>       P_Melody.c                                                   */
/*  <役割>      デモ メロディー関係                                          */
/*  <TAB>       ４タブ編集                                                   */
/*  <保守ﾂｰﾙ>  makefile 参照                                                */
/*  <使用ﾊｰﾄﾞ>  CAT-204-KL5C8012 エーワン(株)                                */
/*                                                                           */
/*****************************************************************************/
#include    <machine.H>
#include    "CAT204.H"
#include    "DemoCtl.h"
/*****************************************************************************/
/*          音階 Hz                                                          */
/*****************************************************************************/
#define dO          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             /* ド */
#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            /* ド */

```

```
34: #define SCMAX       32                  /* 楽譜テーブルの最大数              */
35: /*****************************************************************************/
36: /*      変数宣言                                                             */
37: /*****************************************************************************/
38:         Ushort      MelodyHz;           /* Melody Hz                         */
39:         Uchar       MelMode;            /* 演奏ﾓｰﾄﾞ                          */
40:         Uchar       MelSelect;          /* 自動選曲                          */
41:         Uchar       MelStep;            /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                       */
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:      {250,250,250,250,250,250,250,250,250,250,250,250,1000,
 73:       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                */
 91:     if (MelMode == 0) ManualMelody();   /* 手動演奏                        */
 92:     else              AutoMelody();     /* 自動演奏                        */
 93: }
 94: /*************************************************************************/
 95: /*      MelodySequence()    ﾒﾛﾃﾞｨｰ操作ｺﾝﾄﾛｰﾙ                            */
 96: /*************************************************************************/
 97: void    MelodySequence()
 98: {
 99:     Uchar   port;
100:
101:     port = GetUpPort(1);                /* PB[P30]->PB[P33]                */
102:     if (port & 0xe) {                   /* PB[P31]->PB[P33] ON(立上) ?     */
103:         Buzzer(0);                      /* Buzer OFF                       */
104:         if (port & 0x2) {               /* PB[P31] ON ? ﾓｰﾄﾞ変更           */
105:             MelStep = 0;                /* 強制停止                        */
```

```
106:            if (MelMode == 0) {         /* 現在手動 ?                       */
107:                MelMode   = 1;          /* 自動に変更                       */
108:                MelSelect = 1;          /* 自動選曲                         */
109:            }
110:            else {                      /* 現在自動 ?                       */
111:                MelMode   = 0;          /* 手動に変更                       */
112:                MelSelect = 0;          /* 自動選曲                         */
113:            }
114:        }
115:        if (MelMode != 0) {             /* 自動 ?                           */
116:            if (port & 0x4) {           /* PB[P32] ON ?  選曲               */
117:                MelStep = 0;            /* 強制停止                         */
118:                if (++MelSelect > 3) MelSelect = 1;
119:            }
120:            if (port & 0x8) {           /* PB[P33] ON ? 自動演奏開始        */
121:                if (MelStep == 0) MelStep = 1;      /*  開始               */
122:                else              MelStep = 0;      /*  停止               */
123:                Music   = 0;
124:            }
125:        }
126:        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:     MelodyHz  = 0;                  /* Melody Min Hz                        */
137:     MelMode   = 0;                  /* 演奏ﾓｰﾄﾞ                             */
138:     MelStep   = 0;                  /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                          */
139:     MelSelect = 0;                  /* 自動選曲                             */
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:         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。

８８～９３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。

９７～１３０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。

１３４～１４０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。

１４４～１４７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。

１５１～１９３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 “Cat204p.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  <サンプル>  ポーリング                                                   */
  4: /*                                                                           */
  5: /*  <MOD>       Cat204p.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-204-KL5C8012 エーワン(株)                                 */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.H"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*      外部変数使用                                                         */
 17: /*****************************************************************************/
 18: extern  short      BuzzerHz;          /* TIMER  Buzzer Hz                   */
 19: extern  Uchar      K51Step;           /* USART  ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                 */
 20: extern  Uchar      K51Select;         /*        ﾎﾞｰﾚｰﾄ選択                   */
 21: extern  Uchar      MelStep;           /* Melody ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                 */
 22: extern  Uchar      MelMode;           /*        演奏ﾓｰﾄﾞ                     */
 23: extern  Uchar      MelSelect;         /*        自動選曲                     */
 24: /*****************************************************************************/
 25: /*      変数宣言                                                             */
 26: /*****************************************************************************/
 27:         Uchar      ModeStep;          /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                  */
 28:         Uchar      Shift;             /* shift ﾊﾟﾀｰﾝ                        */
 29: /*****************************************************************************/
 30: /*      main()                                                               */
 31: /*****************************************************************************/
 32: void    main(void)
 33: {
```

```
34:     outp(SCR1,0xc0);                /* SYS    ExtMem 0wait ExtIO 1wait  */
35:     outp(SCR0,0x10);                /*        (OUTBx,OUTAx)             */
36:     SoftWait1ms(20);                /* Power On Wait(20ms) 安定待ち     */
37:     MemInitial();                   /* ﾒﾓﾘ系初期化                       */
38:     IoInitial();                    /* I/O 系初期化                      */
39:     while(1) {
40:         SigInput();                 /* Signal Input  Process             */
41:         SoftWait1ms(20);            /*        ﾎﾟｰﾘﾝｸﾞ用 20ms ﾁｬﾀ取り     */
42:         ModeCntrol();               /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                        */
43:         AllLcdDisp();               /* LCD    全画面表示                 */
44:         SigOutput();                /* Signal Output Process(LED 点灯)  */
45:     }
46: }
47: /*****************************************************************************/
48: /*      Ｍｅｍ初期化                                                         */
49: /*****************************************************************************/
50: void    MemInitial(void)
51: {
52:     ModeStep = 0;                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                 */
53:     Shift = 0;                      /* Led Disp Patan Initial            */
54:
55:     PioMemInitial();                /* PIO    Mem 初期化                 */
56:     LcdMemInitial();                /* LCD    Mem 初期化                 */
57:     TimMemInitial();                /* TIM    Mem 初期化                 */
58:     K51MemInitial();                /* SIO    Mem 初期化                 */
59:     MelMemInitial();                /* Melody Mem 初期化                 */
60: }
61: /*****************************************************************************/
62: /*      Ｉ／Ｏ初期化                                                         */
63: /*****************************************************************************/
64: void    IoInitial(void)
65: {
66:     PioIoInitial();                 /* PIO    I/O 初期化                 */
67:     LcdIoInitial();                 /* LCD    I/O 初期化                 */
68:     TimIoInitial();                 /* TIM    I/O 初期化                 */
69:     K51IoInitial();                 /* SIO    I/O 初期化                 */
```

```
70:     MelIoInitial();                 /* Melody I/O 初期化                */
71: }
72: /*************************************************************************/
73: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                       */
74: /*************************************************************************/
75: void    ModeCntrol()
76: {
77:     if (GetUpPort(1) & 0x1) {           /* PB[P30] ON?(立上)             */
78:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST    */
79:         else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST    */
80:         else if (ModeStep < 30) ModeStep = 30;      /*  USART Goto TEST    */
81:         else if (ModeStep < 40) ModeStep = 40;      /*  Demo  Melody       */
82:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ     */
83:         Buzzer(0);                                  /*  強制 OFF           */
84:     }
85:     switch(ModeStep) {
86:     case 0:
87:         GotoxyMemSet(0,0,"CAT204&BF3000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ      */
88:         GotoxyMemSet(0,1,"Polling    [P30]");
89:         ModeStep++;
90:         break;
91:     case 1:
92:         RunRun();                       /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ     */
93:         break;
94:     case 10:                            /*  PIO   TEST                    */
95:         GotoxyMemSet(0,0,"PIO             ");
96:         GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
97:         ModeStep++;
98:         break;
99:     case 11:
100:        PioDemo();
101:        break;
102:    case 20:                            /*  Timer TEST                    */
103:        GotoxyMemSet(0,0,"Timer/Counter   ");
104:        GotoxyMemSet(0,1,"0000Hz[+P33-P32]");
105:        BuzzerHz = 0;                           /* Buzzer Hz              */
```

```
106:          ModeStep++;
107:          break;
108:      case 21:
109:          TimerDemo();
110:          RunRun();                          /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ        */
111:          break;
112:      case 30:                               /*  USART TEST                      */
113:          GotoxyMemSet(0,0,"Usart 8,n,1[P31]");
114:          GotoxyMemSet(0,1,"09600b[+P33-P32]");
115:          K51Step   = 0;                                 /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ          */
116:          K51Select = 2;                                 /* 9600BPS             */
117:          ModeStep++;
118:          break;
119:      case 31:
120:          K51Demo();
121:          RunRun();                          /* ｼﾌﾄ LED 点灯 OUT ﾊﾞｯﾌｧｰにｾｯﾄ        */
122:          break;
123:      case 40:                               /*  Demo  Melody                   */
124:          GotoxyMemSet(0,0,"Melody          ");
125:          GotoxyMemSet(0,1,"P31[M]33[ｽ]32[ｾ]");
126:          MelStep   = 0;                                 /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ          */
127:          MelMode   = 0;                                 /* 演奏ﾓｰﾄﾞ             */
128:          MelSelect = 0;                                 /* 自動選曲             */
129:          ModeStep++;
130:          break;
131:      case 41:
132:          Melody();
133:          RunRun();
134:          break;
135:      }
136: }
137:
138: /*************************************************************************/
139: /*      RunRun()        CPU 走行表示                                     */
140: /*************************************************************************/
141: void    RunRun()
```

```
142: {
143:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
144:     PutOutPort(Shift,'=');
145: }
146: /*****************************************************************************/
147: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                     */
148: /*****************************************************************************/
149: void    SoftWait1ms(Ushort ms)
150: {
151:     while(ms-- != 0) {
152:         Wait1ms();
153:     }
154: }
155: /*****************************************************************************/
156: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                    */
157: /*****************************************************************************/
158: void    Wait1ms()
159: {
160:     _asm_("\n           PUSH    HL              \n");
161:     _asm_("\n           LD      HL,1228         \n");   /* 1228*6=7372cyc   */
162:     _asm_("\n W01:                              \n");
163:     _asm_("\n           DEC     HL              \n");   /* cyc = 1          */
164:     _asm_("\n           LD      A,L             \n");   /*     = 1          */
165:     _asm_("\n           OR      H               \n");   /*     = 1          */
166:     _asm_("\n           JP      NZ,W01          \n");   /*     = 3          */
167:     _asm_("\n           POP     HL              \n");   /*    += 6          */
168: }
169: /*****************************************************************************/
170: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                    */
171: /*****************************************************************************/
172: void    SoftWait10us(Ushort us)
173: {
174:     while(us-- != 0) {
175:         Wait10us();
176:     }
177: }
```

```
178: /*****************************************************************************/
179: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                    */
180: /*****************************************************************************/
181: void    Wait10us()
182: {
183:     _asm_("\n           PUSH    HL              \n");
184:     _asm_("\n           LD      HL,13           \n");   /* 13*6=78cyc       */
185:     _asm_("\n W02:                              \n");
186:     _asm_("\n           DEC     HL              \n");   /* cyc = 1          */
187:     _asm_("\n           LD      A,L             \n");   /*     = 1          */
188:     _asm_("\n           OR      H               \n");   /*     = 1          */
189:     _asm_("\n           JP      NZ,W02          \n");   /*     = 3          */
190:     _asm_("\n           POP     HL              \n");   /*    += 6          */
191: }
```

［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。

２１～２３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。

５９行：

“Ｐ_Ｍｅｌｏｄｙ．ｃ”で使用する変数の初期化関数を呼んでいます。

７０行：

“Ｐ_Ｍｅｌｏｄｙ．ｃ”で使用するＩ／Ｏを初期化する関数を呼んでいます。

１２３～１３４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。

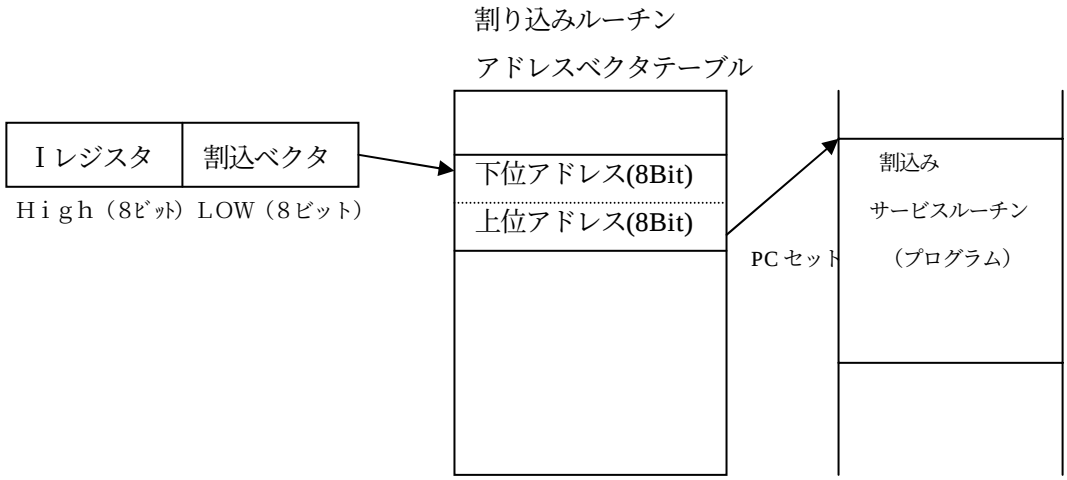
第２部　割込み編
第１章　タイマ割込み

　ここの章では、割り込みを使用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。
又、割り込み要因として、タイマＢチャネル２の連続カウントモードでの１０ｍｓ毎割り込み例を使用します。
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￥
にディレクトリの移動をしておいて下さい。

１．スタートアップを割り込み用に変更する。

　ＫＣ８０で割り込みを使用する場合は、割り込み要因配列のきまった割り込みベクター方式を使用しています。
これは、ＫＣ８０内部の割り込みコントローラ仕様によるものです。

概略仕様
- クロック同期式の割り込みコントローラ
- モード２割り込み対応
- １６レベルの割り込み要因に対し、優先順位の制御ができる
- １６レベル個々にマスクができる
- 多重割り込みが可能
- 割り込み要求入力のエッジ／レベルの選択ができる
- 不正割り込み検出機能がある
- 割り込み処理終了は、ＲＥＴＩ命令で検出する。



図［2・1－1－1］　モード2割り込みシーケンス

レベル名	割り込み要求元
ＩＲ［　0］	外部入力Ｐ００
ＩＲ［　1］	外部入力Ｐ０１
ＩＲ［　2］	外部入力Ｐ０２
ＩＲ［　3］	外部入力Ｐ０３
ＩＲ［　4］	外部入力Ｐ０４
ＩＲ［　5］	外部入力Ｐ０５
ＩＲ［　6］	外部入力Ｐ０６
ＩＲ［　7］	外部入力Ｐ０７
ＩＲ［　8］	ＵＳＡＲＴ　ＴＸＲＤＹ
ＩＲ［　9］	ＵＳＡＲＴ　ＲＸＲＤＹ
ＩＲ［１０］	ＵＳＡＲＴ　ＴＸＥＭＰＴＹ
ＩＲ［１１］	タイマ／カウンタ　Ａ　ＣＨ０
ＩＲ［１２］	タイマ／カウンタ　Ａ　ＣＨ１
ＩＲ［１３］	タイマ／カウンタ　Ｂ　ＣＨ０　ＯＵＴＳ
ＩＲ［１４］	タイマ／カウンタ　Ｂ　ＣＨ１　ＯＵＴＳ
ＩＲ［１５］	タイマ／カウンタ　Ｂ　ＣＨ２　ＯＵＴＳ

図［2・1－1－2］　割り込みコントローラの要因とアドレスベクタテーブル

1）プログラムリスト

file “StartupC.asm”

```
  1: ;/*****************************************************************************/
  2: ;/*      <MOD>     StartupC.asm                                               */
  3: ;/*      <役割>    スタートアップ (割込み対応 CAT204-KL5C8012)                */
  4: ;/*                リンクの都合上、先頭に指定することが絶対条件               */
  5: ;/*                $$Rx_init_value_は、knil（リンク）で作成されます。         */
  6: ;/*      <TAB>     ４タブ編集                                                 */
  7: ;/*****************************************************************************/
  8: STACK       EQU     0xFFFF          ; ｽﾀｯｸﾎﾞﾄﾑ
  9: BBR1        EQU     0x0             ; KL5C8012(MMU) BBR1
 10: BR1         EQU     0x1             ;               BR1
 11: BBR2        EQU     0x2             ;               BBR2
 12: BR2         EQU     0x3             ;               BR2
 13: BBR3        EQU     0x4             ;               BBR3
 14: BR3         EQU     0x5             ;               BR3
 15: BBR4        EQU     0x6             ;               BBR4
 16: BR4         EQU     0x7             ;               BR4
 17: ;/*****************************************************************************/
 18: ;/*          スタートアップ                                                   */
 19: ;/*****************************************************************************/
 20:             CSEG
 21: StartUp:
 22:             ORG     0
 23:             DI
 24:             LD      SP,STACK
 25:             IM      2
 26:             LD      A,0
 27:             LD      I,A
 28: ;/*****************************************************************************/
 29: ;/*          MMUの初期設定                                                    */
 30: ;/*****************************************************************************/
 31:             LD      A,LOW  $$R1_init_value_##
 32:             OUT     (BBR1),A
 33:             LD      A,HIGH $$R1_init_value_##
```

```
34:             OUT     (BR1),A
35:             LD      A,LOW  $$R2_init_value_##
36:             OUT     (BBR2),A
37:             LD      A,HIGH $$R2_init_value_##
38:             OUT     (BR2),A
39:             LD      A,LOW  $$R3_init_value_##
40:             OUT     (BBR3),A
41:             LD      A,HIGH $$R3_init_value_##
42:             OUT     (BR3),A
43:             LD      A,LOW  $$R4_init_value_##
44:             OUT     (BBR4),A
45:             JP      main_##
46: ;/*****************************************************************************/
47: ;/*          割込みベクター                                                   */
48: ;/*****************************************************************************/
49:             ORG     0x80
50:             DW      NON             ; IR[ 0] P00
51:             DW      NON             ; IR[ 1] P01
52:             DW      NON             ; IR[ 2] P02
53:             DW      NON             ; IR[ 3] P03
54:             DW      NON             ; IR[ 4] P04
55:             DW      NON             ; IR[ 5] P05
56:             DW      NON             ; IR[ 6] P06(IRQ1)
57:             DW      NON             ; IR[ 7] P07(IRQ2)
58:             DW      NON             ; IR[ 8] TXRDY
59:             DW      NON             ; IR[ 9] RXRDY
60:             DW      NON             ; IR[10] TXEMPTY
61:             DW      NON             ; IR[11] TIMER A CH0
62:             DW      NON             ; IR[12] TIMER A CH1
63:             DW      NON             ; IR[13] TIMER B CH0 OUTS
64:             DW      NON             ; IR[14] TIMER B CH1 OUTS
65:             DW      TIMERB2         ; IR[15] TIMER B CH2 OUTS
66: ;/*****************************************************************************/
67: ;/*          割込みハンドラー                                                 */
68: ;/*****************************************************************************/
69: NON::                               ; 割込み未使用
```

```
70:             EI
71:             RETI
72: TIMERB2::                           ; IR[15] TIMER B CH2 OUTS
73:             PUSH    AF
74:             PUSH    BC
75:             PUSH    DE
76:             PUSH    HL
77:             PUSH    IX
78:             PUSH    IY
79:             CALL    TimerB2_##
80:             POP     IY
81:             POP     IX
82:             POP     HL
83:             POP     DE
84:             POP     BC
85:             POP     AF
86:             EI
87:             RETI
88: ;/*************************************************************************/
89: ;/*          終了（C ソースへのエントリー）                                 */
90: ;/*************************************************************************/
91:             ORG     0x200
92:             END
```

［リストの説明］

２５行：

割り込みモード２だとＣＰＵに教える命令です。

２６～２７行：

図［２・１－１－１］の説明にもあるように、割り込みベクターテーブルアドレスの上位８ビットをＩレジにセットします。

このサンプルは、０ｘ８０番地がベクターテーブルアドレスにしていますので、ゼロ（０）をセットしています。

４９～６５行：

図［２・１－１－２］の割り込み要因ごとの割り込みベクターテーブルです。

割り込みが発生した場合、このテーブルに登録されたアドレス値にプログラムはジャンプします。

未使用割り込みは、とりあえず“ＮＯＮ：”という割り込みハンドラを登録しましたが、現実にはあり得ないことです。

６５行目は、今回組み込む“タイマＢチャネル２のＯＵＴＳ出力”の割り込みハンドラを登録しました。

６９～７１行：

未使用割り込みハンドラです。

７２～８７行：

タイマＢチャネル２のＯＵＴＳ出力の割り込みハンドラです。

全レジスタのＰＵＳＨをしてから、Ｃ言語記述の“ＴｉｍｅｒＢ２（）”を呼び、戻って来てから全レジスタのＰＯＰを実行し、“ＥＩ”と“ＲＥＴＩ”で終了しています。

多重割り込み処理をしたい場合は、全レジスタのＰＵＳＨ後に“ＥＩ”命令を置きますと、実現されます。（スタック領域にご注意！！）

これが、Ｃ言語用割り込みハンドラの雛型です。

２．タイマモジュールを割り込み用に変更する。

タイマＢチャネル２の連続カウントモードで１０ｍｓ毎割り込みを使用する場合、タイマＢのモード変更が必要になります。
また、割り込みハンドラから呼ばれる関数“ＴｉｍｅｒＢ２”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。

プログラムに沿って説明します。

１）プログラムリスト

file “I_Time.c”

```
/************************************************************************/
/*                                                                      */
/*  ＜サンプル＞  割込み                                                */
/*                                                                      */
/*  ＜MOD＞       I_Time.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-204-KL5C8012 エーワン(株)                         */
/*                                                                      */
/************************************************************************/
#include    <machine.H>
#include    "CAT204.H"
#include    "DemoCtl.h"
/************************************************************************/
/*      マクロ宣言                                                      */
/************************************************************************/
#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                */
```

```
25:         Uchar       TmSt[TMMAX];        /*          Start ﾌﾗｸﾞ           */
26:         Ushort      TmCnt[TMMAX];       /*          Count               */
27: /*************************************************************************/
28: /*      Ｍｅｍ初期化                                                     */
29: /*************************************************************************/
30: void    TimMemInitial(void)
31: {
32:     BuzzerHz = 0;                       /* Buzzer Hz                    */
33:     memset(TmSt, OFF,sizeof(TmSt));     /* Timer(10ms) Initial          */
34:     memset(TmUp, OFF,sizeof(TmUp));
35:     memset(TmCnt,OFF,sizeof(TmCnt));
36: }
37: /*************************************************************************/
38: /*      Ｉ／Ｏ初期化                                                     */
39: /*************************************************************************/
40: void    TimIoInitial(void)
41: {
42:     outp(TCWDA0,0);                     /* TIMA TM0 未使用              */
43:     outp(TCWDA1,0);                     /*      TM1 未使用              */
44:     outp(TCWDB0,0);                     /* TIMB TM0 使用   Buzzer       */
45:     outp(TCWDB1,0);                     /*      TM1 使用   USART ﾎﾞｰﾚｰﾄ  */
46:     outp(TCWDB2,0x4);                   /*      TM2 使用   10ms 割込み  */
47:     outp(TCNTB2,(288 & 0xFF));          /*          outL+連続+sys/256   */
48:     outp(TCNTB2,(288 >> 8));            /*          7,372,800/256=28800Hz */
49:                                         /*          28800/288    =100Hz */
50: }
51: /*************************************************************************/
52: /*      Buzzer   Timer B ﾁｬﾝﾈﾙ 0  OUTBP0 出力に Buzzer 接続              */
53: /*************************************************************************/
54: void    Buzzer(Ushort hz)
55: {
56:     Uchar   cyc;
57:
58:     if ((hz >= BZMIN) && (hz <= BZMAX)) {   /* Buzzer ON                */
59:         outp(TCWDB0,0x1c);              /* outH+PWM+sys/256=28800Hz     */
60:         cyc = 28800 / hz;               /* 周期の計算                   */
```

```
61:         outp(TCNTB0,cyc-1);             /* 周期設定                    */
62:         outp(TCNTB0,(cyc/2)-1);         /* 巾設定                      */
63:     }
64:     else {                              /* Buzzer OFF                  */
65:         outp(TCWDB0,0);
66:         outp(TCNTB0,0);
67:         outp(TCNTB0,0);
68:     }
69: }
70: /*****************************************************************************/
71: /*      TmStart                                                              */
72: /*****************************************************************************/
73: void    TmStart(short tno,short ms)
74: {
75:     Ushort  cnt;
76:
77:     cnt = ms / 10;                      /* 単位 10ms に修正する        */
78:     TmUp[tno]  = OFF;
79:     TmCnt[tno] = cnt;
80:     TmSt[tno]  = ON;
81: }
82: /*****************************************************************************/
83: /*      TmUpTest    ﾀｲﾏ UP ﾌﾗｸﾞ ﾃｽﾄ                                          */
84: /*****************************************************************************/
85: Uchar    TmUpTest(short tno)
86: {
87:     return(TmUp[tno]);
88: }
89: /*****************************************************************************/
90: /*      ＴＭＢ２　タイマー割り込み（10ms）　割り込みハンドラより呼ばれる    */
91: /*****************************************************************************/
92: void    TimerB2(void)
93: {
94:     short   i;
95:
96:     for(i = 0;i < TMMAX;i++) {          /*  内部タイマー処理           */
```

```
 97:         if (TmSt[i] == ON) {
 98:             if (--TmCnt[i] == 0) {
 99:                 TmUp[i] = ON;
100:                 TmSt[i] = OFF;
101:             }
102:         }
103:     }
104: }
105: /*************************************************************************/
106: /*      TimerDemo   Timer デモ                                           */
107: /*************************************************************************/
108: void    TimerDemo()
109: {
110:     Uchar   port;
111:     Uchar   dec[4+1];
112:
113:     port = GetUpPort(1);                /* PB[P30]->PB[P33]             */
114:     if (port & 0xc) {                   /* PB[P32] | PB[P33] ON ?       */
115:         if (port & 0x4) {               /* PB[-P32] ON-立上り            */
116:             BuzzerHz -= 100;
117:         }
118:         else if (port & 0x8) {          /* PB[+P33] ON-立上り            */
119:             BuzzerHz += 100;
120:         }
121:         if (BuzzerHz < BZMIN) BuzzerHz = BZMIN;
122:         if (BuzzerHz > BZMAX) BuzzerHz = BZMAX;
123:         Buzzer(BuzzerHz);
124:         Bin2AdecN(dec,BuzzerHz,4);      /* 表示用データ作成             */
125:         GotoxyMemSet(0,1,dec);
126:     }
127: }
```

［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。

４６～５０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、タイマＢのチャネル２の初期化をします。　　図［５－２－４］参照

ＣＯＵＮＴＣＬＫ：　システムクロックの２５６分周　“００”

ＣＯＵＮＴＭＯＤＥ：　連続カウントモード　　“０１”

ＴＯＧＧＬＥ：　ＯＵＴＰ端子の初期を“Ｌ”　　“０”

［１０ｍｓ［１００Ｈｚ］タイミングの計算］

システムクロック　÷　分周　　＝　タイマ基本クロック

（７３７２８００）　（２５６）　　（２８８００）Ｈｚ

タイマ基本クロック　÷　タイマ出力クロック　＝　カウント値

（２８８００）　　（１００）　　（２８８）

になり、タイマＢチャネル２のカウント値は、“２８８”になります。

７３～８１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。

８５～８８行：

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

９２～１０４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。

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

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と割り込みコントローラへの設定をしています。

プログラムに沿って説明します。

１）プログラムリスト

file “Cat204i1.c”

```
/*****************************************************************************/
/*                                                                           */
/*  ＜サンプル＞ 割込み                                                      */
/*                                                                           */
/*  ＜MOD＞      Cat204i1.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-204-KL5C8012 エーワン(株)                                */
/*                                                                           */
/*****************************************************************************/
#include    <machine.H>
#include    "CAT204.H"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      外部変数使用                                                         */
/*****************************************************************************/
extern  short       BuzzerHz;           /* TIMER  Buzzer Hz                  */
extern  Uchar       K51Step;            /* USART  ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                */
extern  Uchar       K51Select;          /*        ﾎﾞｰﾚｰﾄ選択                 */
extern  Uchar       MelStep;            /* Melody 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                */
        Uchar       Shift;              /* shift ﾊﾟﾀｰﾝ                        */
/*****************************************************************************/
/*      main()                                                               */
/*****************************************************************************/
void    main(void)
{
    outp(SCR1,0xc0);                    /* SYS    ExtMem 0wait ExtIO 1wait  */
    outp(SCR0,0x10);                    /*        (OUTBx,OUTAx)             */
    SoftWait1ms(20);                    /* Power On Wait(20ms) 安定待ち      */
    MemInitial();                       /* ﾒﾓﾘ系初期化                       */
    IoInitial();                        /* I/O 系初期化                      */
    TmStart(TM0,20);                    /* ﾁｬﾀﾘﾝｸﾞ防止ｽﾀｰﾄ                   */
    ei();                               /* <- 割り込み許可                   */
    while(1) {
        if (TmUpTest(TM0) == ON) {
            TmStart(TM0,20);            /* ﾁｬﾀﾘﾝｸﾞ防止ｽﾀｰﾄ                   */
            SigInput();                 /* Signal Input  Process             */
            ModeCntrol();               /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                        */
            SigOutput();                /* Signal Output Process             */
        }
        AllLcdDisp();                   /* LCD    全画面表示                 */
    }
}
/*****************************************************************************/
/*      Ｍｅｍ初期化                                                         */
/*****************************************************************************/
void    MemInitial(void)
{
    ModeStep = 0;                       /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
    Shift = 0;                          /* Led Disp Patan Initial            */

    PioMemInitial();                    /* PIO    Mem 初期化                 */
    LcdMemInitial();                    /* LCD    Mem 初期化                 */
    TimMemInitial();                    /* TIM    Mem 初期化                 */
```

```
62:     K51MemInitial();                    /* SIO    Mem 初期化              */
63:     MelMemInitial();                    /* Melody Mem 初期化              */
64: }
65: /*****************************************************************************/
66: /*       Ｉ／Ｏ初期化                                                        */
67: /*****************************************************************************/
68: void    IoInitial(void)
69: {
70:     PioIoInitial();                     /* PIO    I/O 初期化              */
71:     LcdIoInitial();                     /* LCD    I/O 初期化              */
72:     TimIoInitial();                     /* TIM    I/O 初期化              */
73:     K51IoInitial();                     /* SIO    I/O 初期化              */
74:     MelIoInitial();                     /* Melody I/O 初期化              */
75:
76:     outp(LERL,0);                       /* 割込<- ﾚﾍﾞﾙ/ｴｯｼﾞ設定           */
77:     outp(LERH,0x80);                    /*      <- IR[15]TIMER B CH2 OUTSｴｯｼﾞ*/
78:     outp(IVR,0x80);                     /*      <- 先頭ﾍﾞｸﾀ 80H           */
79:     outp(PGRL,0);                       /*      <- ﾌﾟﾗｲｵﾘﾃｨの設定          */
80:     outp(PGRH,0x80);                    /*      <- IR[15]TIMER B CH2 OUTS  */
81:     outp(IMRL,0xFF);                    /*      <- ﾏｽｸﾚｼﾞｽﾀ                */
82:     outp(IMRH,0x7F);                    /*      <- IR[15]TIMER B CH2 OUTS  */
83: }
84: /*****************************************************************************/
85: /*       ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                          */
86: /*****************************************************************************/
87: void    ModeCntrol()
88: {
89:     if (GetUpPort(1) & 0x1) {           /* PB[P30] ON?(立上)              */
90:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST   */
91:         else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST   */
92:         else if (ModeStep < 30) ModeStep = 30;      /*  USART Goto TEST   */
93:         else if (ModeStep < 40) ModeStep = 40;      /*  Demo  Melody      */
94:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ     */
95:         Buzzer(0);                                  /*  強制 OFF          */
96:     }
97:     switch(ModeStep) {
```

```
 98:     case 0:
 99:         GotoxyMemSet(0,0,"CAT204&BF3000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ       */
100:         GotoxyMemSet(0,1,"Interrupt  [P30]");
101:         ModeStep++;
102:         break;
103:     case 1:
104:         RunRun();
105:         break;
106:     case 10:                                        /*  PIO   TEST          */
107:         GotoxyMemSet(0,0,"PIO              ");
108:         GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
109:         ModeStep++;
110:         break;
111:     case 11:
112:         PioDemo();
113:         break;
114:     case 20:                                        /*  Timer TEST          */
115:         GotoxyMemSet(0,0,"Timer/Counter   ");
116:         GotoxyMemSet(0,1,"0000Hz[+P33-P32]");
117:         BuzzerHz = 0;                               /* Buzzer Hz            */
118:         ModeStep++;
119:         break;
120:     case 21:
121:         TimerDemo();
122:         RunRun();
123:         break;
124:     case 30:                                        /*  USART TEST          */
125:         GotoxyMemSet(0,0,"Usart 8,n,1[P31]");
126:         GotoxyMemSet(0,1,"09600b[+P33-P32]");
127:         K51Step   = 0;                              /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ          */
128:         K51Select = 2;                              /* 9600BPS              */
129:         ModeStep++;
130:         break;
131:     case 31:
132:         K51Demo();
133:         RunRun();
```

```
134:            break;
135:        case 40:                                        /*  Demo  Melody        */
136:            GotoxyMemSet(0,0,"Melody          ");
137:            GotoxyMemSet(0,1,"P31[M]33[ｽ]32[ｾ]");
138:            MelStep   = 0;                              /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ          */
139:            MelMode   = 0;                              /* 演奏ﾓｰﾄﾞ             */
140:            MelSelect = 0;                              /* 自動選曲             */
141:            ModeStep++;
142:            break;
143:        case 41:
144:            Melody();
145:            RunRun();
146:            break;
147:        }
148: }
149: /*****************************************************************************/
150: /*      RunRun()        CPU 走行表示                                         */
151: /*****************************************************************************/
152: void    RunRun()
153: {
154:     if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
155:     PutOutPort(Shift,'=');
156: }
157: /*****************************************************************************/
158: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                    */
159: /*****************************************************************************/
160: void    SoftWait1ms(Ushort ms)
161: {
162:     while(ms-- != 0) {
163:         Wait1ms();
164:     }
165: }
166: /*****************************************************************************/
167: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                   */
168: /*****************************************************************************/
169: void    Wait1ms()
```

```
170: {
171:     _asm_("\n          PUSH    HL              \n");
172:     _asm_("\n          LD      HL,1228         \n");   /* 1228*6=7372cyc   */
173:     _asm_("\n W01:                             \n");
174:     _asm_("\n          DEC     HL              \n");   /* cyc = 1          */
175:     _asm_("\n          LD      A,L             \n");   /*     = 1          */
176:     _asm_("\n          OR      H               \n");   /*     = 1          */
177:     _asm_("\n          JP      NZ,W01          \n");   /*     = 3          */
178:     _asm_("\n          POP     HL              \n");   /*    += 6          */
179: }
180: /*************************************************************************/
181: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                */
182: /*************************************************************************/
183: void    SoftWait10us(Ushort us)
184: {
185:     while(us-- != 0) {
186:         Wait10us();
187:     }
188: }
189: /*************************************************************************/
190: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait               */
191: /*************************************************************************/
192: void    Wait10us()
193: {
194:     _asm_("\n          PUSH    HL              \n");
195:     _asm_("\n          LD      HL,13           \n");   /* 13*6=78cyc       */
196:     _asm_("\n W02:                             \n");
197:     _asm_("\n          DEC     HL              \n");   /* cyc = 1          */
198:     _asm_("\n          LD      A,L             \n");   /*     = 1          */
199:     _asm_("\n          OR      H               \n");   /*     = 1          */
200:     _asm_("\n          JP      NZ,W02          \n");   /*     = 3          */
201:     _asm_("\n          POP     HL              \n");   /*    += 6          */
202: }
```

［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。

３９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。

４０行：

ここで、割り込み許可“ｅｉ（）”をします。

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。

４２～４３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。

７６～８２行：

割り込みコントローラへの設定部分です。

アドレス	ブロック名	ライト時	リード時	シンボル
３４Ｈ	割り込みコントローラ	ＬＥＲＬ／ＰＧＲＬ	ＩＳＲＬ	ＬＥＲＬ／ＰＧＲＬ／ＩＳＲＬ
３５Ｈ		ＬＥＲＨ／ＰＧＲＨ	ＩＳＲＨ	ＬＥＲＨ／ＰＧＲＨ／ＩＳＲＨ
３６Ｈ		ＩＭＲＬ	ＩＭＲＬ	ＩＭＲＬ
３７Ｈ		ＩＶＲ／ＩＭＲＨ	ＩＭＲＨ	ＩＶＲ／ＩＭＲＨ

図［１－３－１］　割り込みコントローラＩ／Ｏマップ

［７６～７７行：］

ａ．割り込みのレベル／エッジの設定

ＬＥＲ［１５］＝タイマＢチャネル２ＯＵＴＳをエッジモードにします。

レベル／エッジ　レジスタ（ＬＥＲＨ）　０＝レベル　１＝エッジ　ＷＲＩＴＥ			
ﾋﾞｯﾄ	名称		要求元
7	ＬＥＲ［１５］	ＩＲ［１５］	タイマ／カウンタ　Ｂ　ＣＨ２　ＯＵＴＳ
6	ＬＥＲ［１４］	ＩＲ［１４］	タイマ／カウンタ　Ｂ　ＣＨ１　ＯＵＴＳ
5	ＬＥＲ［１３］	ＩＲ［１３］	タイマ／カウンタ　Ｂ　ＣＨ０　ＯＵＴＳ
4	ＬＥＲ［１２］	ＩＲ［１２］	タイマ／カウンタ　Ａ　ＣＨ１
3	ＬＥＲ［１１］	ＩＲ［１１］	タイマ／カウンタ　Ａ　ＣＨ０
2	ＬＥＲ［１０］	ＩＲ［１０］	ＵＳＡＲＴ　ＴＸＥＭＰＴＹ
1	ＬＥＲ［　９］	ＩＲ［　９］	ＵＳＡＲＴ　ＲＸＲＤＹ
0	ＬＥＲ［　８］	ＩＲ［　８］	ＵＳＡＲＴ　ＴＸＲＤＹ

図［1－3－2］ＬＥＲＨ（Ｌｅｖｅｌ／Ｅｄｇｅ　Ｒｅｇｉｓｔｅｒ）

レベル／エッジ　レジスタ（ＬＥＲＬ）　０＝レベル　１＝エッジ　ＷＲＩＴＥ			
ﾋﾞｯﾄ	名称		要求元
7	ＬＥＲ［ 7］	ＩＲ［ 7］	外部入力Ｐ０７
6	ＬＥＲ［ 6］	ＩＲ［ 6］	外部入力Ｐ０６
5	ＬＥＲ［ 5］	ＩＲ［ 5］	外部入力Ｐ０５
4	ＬＥＲ［ 4］	ＩＲ［ 4］	外部入力Ｐ０４
3	ＬＥＲ［ 3］	ＩＲ［ 3］	外部入力Ｐ０３
2	ＬＥＲ［ 2］	ＩＲ［ 2］	外部入力Ｐ０２
1	ＬＥＲ［ 1］	ＩＲ［ 1］	外部入力Ｐ０１
0	ＬＥＲ［ 0］	ＩＲ［ 0］	外部入力Ｐ００

図［1－3－2］ＬＥＲＬ（Ｌｅｖｅｌ／Ｅｄｇｅ　Ｒｅｇｉｓｔｅｒ）

［７８行：］

ｂ．割り込みベクターアドレスの下位８ビット中の上位３ビットの設定

割り込みベクター値を０ｘ８０にします。

	Ｄ７	Ｄ６	Ｄ５	Ｄ４	Ｄ３	Ｄ２	Ｄ１	Ｄ０
IVR	ＩＶＲ［7］	ＩＶＲ［6］	ＩＶＲ［5］					

図［1－3－3］ＩｎｔｅｒｒｕｐｔＶｅｃｔｏｒＲｅｇｉｓｔｅｒ

	Ｄ７	Ｄ６	Ｄ５	Ｄ４	Ｄ３	Ｄ２	Ｄ１	Ｄ０
IVR	ＩＶＲ［7］	ＩＶＲ［6］	ＩＶＲ［5］					0

割り込み要求（ＩＲ［n］）番号に応じた２進数コード出力

＋ＩＲ［15］	1	1	1	1
＋ＩＲ［14］	1	1	1	0
｜				
｜				
｜				
＋ＩＲ［ 0］	0	0	0	0

図［1－3－4］　割り込みベクタ出力

［７９～８０行：］

ｃ．割り込みプライオリティの設定

ＰＧＲ［１５］＝タイマBチャネル２ＯＵＴＳを“ＨＩＧＨ”グループとします。

プライオリティ　グループ　レジスタ（ＰＧＲＨ）　０＝ＬＯＷ　１＝ＨＩＧＨ　ＷＲＩＴＥ			
ﾋﾞｯﾄ	名称		要求元
7	ＰＧＲ［１５］	ＩＲ［１５］	タイマ／カウンタ　Ｂ　ＣＨ２　ＯＵＴＳ
6	ＰＧＲ［１４］	ＩＲ［１４］	タイマ／カウンタ　Ｂ　ＣＨ１　ＯＵＴＳ
5	ＰＧＲ［１３］	ＩＲ［１３］	タイマ／カウンタ　Ｂ　ＣＨ０　ＯＵＴＳ
4	ＰＧＲ［１２］	ＩＲ［１２］	タイマ／カウンタ　Ａ　ＣＨ１
3	ＰＧＲ［１１］	ＩＲ［１１］	タイマ／カウンタ　Ａ　ＣＨ０
2	ＰＧＲ［１０］	ＩＲ［１０］	ＵＳＡＲＴ　ＴＸＥＭＰＴＹ
1	ＰＧＲ［　９］	ＩＲ［　９］	ＵＳＡＲＴ　ＲＸＲＤＹ
0	ＰＧＲ［　８］	ＩＲ［　８］	ＵＳＡＲＴ　ＴＸＲＤＹ

プライオリティ　グループ　レジスタ（ＰＧＲＬ）　０＝ＬＯＷ　１＝ＨＩＧＨ　ＷＲＩＴＥ			
ﾋﾞｯﾄ	名称		要求元
7	ＰＧＲ［　７］	ＩＲ［　７］	外部入力Ｐ０７
6	ＰＧＲ［　６］	ＩＲ［　６］	外部入力Ｐ０６
5	ＰＧＲ［　５］	ＩＲ［　５］	外部入力Ｐ０５
4	ＰＧＲ［　４］	ＩＲ［　４］	外部入力Ｐ０４
3	ＰＧＲ［　３］	ＩＲ［　３］	外部入力Ｐ０３
2	ＰＧＲ［　２］	ＩＲ［　２］	外部入力Ｐ０２
1	ＰＧＲ［　１］	ＩＲ［　１］	外部入力Ｐ０１
0	ＰＧＲ［　０］	ＩＲ［　０］	外部入力Ｐ００

図［1－3－5］　ＰｒｉｏｒｉｔｙＧｒｏｕｐＲｅｇｉｓｔａｅｒ

［８１～８２行：］

ｄ．割り込みマスクの設定（０＝非マスク状態：割込み可　１＝マスク状態：割込み不可）

ＩＭＲ［１５］＝タイマＢチャネル２ＯＵＴＳのマスクを解除します。

インタラプト　マスク　レジスタ（ＩＭＲＨ）　０＝非マスク状態　１＝マスク状態　ＷＲＩＴＥ			
ﾋﾞｯﾄ	名称		要求元
7	ＩＭＲ［１５］	ＩＲ［１５］	タイマ／カウンタ　Ｂ　ＣＨ２　ＯＵＴＳ
6	ＩＭＲ［１４］	ＩＲ［１４］	タイマ／カウンタ　Ｂ　ＣＨ１　ＯＵＴＳ
5	ＩＭＲ［１３］	ＩＲ［１３］	タイマ／カウンタ　Ｂ　ＣＨ０　ＯＵＴＳ
4	ＩＭＲ［１２］	ＩＲ［１２］	タイマ／カウンタ　Ａ　ＣＨ１
3	ＩＭＲ［１１］	ＩＲ［１１］	タイマ／カウンタ　Ａ　ＣＨ０
2	ＩＭＲ［１０］	ＩＲ［１０］	ＵＳＡＲＴ　ＴＸＥＭＰＴＹ
1	ＩＭＲ［　９］	ＩＲ［　９］	ＵＳＡＲＴ　ＲＸＲＤＹ
0	ＩＭＲ［　８］	ＩＲ［　８］	ＵＳＡＲＴ　ＴＸＲＤＹ

インタラプト　マスク　レジスタ（ＩＭＲＨ）　０＝非マスク状態　１＝マスク状態　ＷＲＩＴＥ			
ﾋﾞｯﾄ	名称		要求元
7	ＩＭＲ［　７］	ＩＲ［　７］	外部入力Ｐ０７
6	ＩＭＲ［　６］	ＩＲ［　６］	外部入力Ｐ０６
5	ＩＭＲ［　５］	ＩＲ［　５］	外部入力Ｐ０５
4	ＩＭＲ［　４］	ＩＲ［　４］	外部入力Ｐ０４
3	ＩＭＲ［　３］	ＩＲ［　３］	外部入力Ｐ０３
2	ＩＭＲ［　２］	ＩＲ［　２］	外部入力Ｐ０２
1	ＩＭＲ［　１］	ＩＲ［　１］	外部入力Ｐ０１
0	ＩＭＲ［　０］	ＩＲ［　０］	外部入力Ｐ００

図［1－3－6］　ＩｎｔｅｒｒｕｐｔＭａｓｋＲｅｇｉｓｔｅｒ

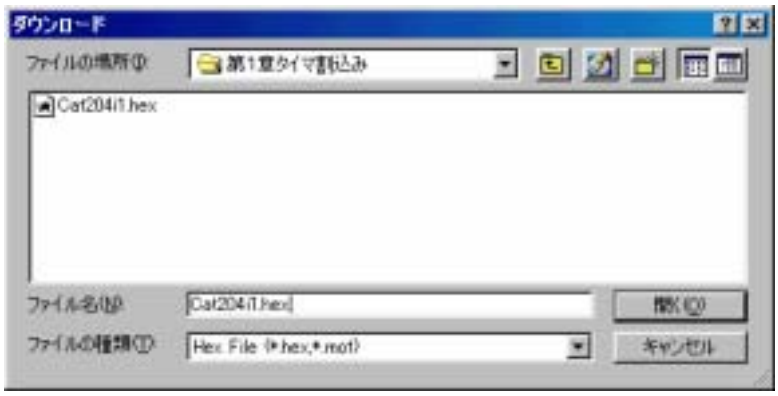
８４～最終行：

変更なし。

以上、３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。

ＡＢＣｗｉｎでダウンロード後、プログラム実行をして下さい。



“Ｃａｔ２０４ｉ１．ｈｅｘ”です

次章は、ＵＳＡＲＴ割り込みに挑戦したいと思います。

第2章　ＵＳＡＲＴ割込み

　ここの章では、ＵＳＡＲＴ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）

¥評価ボード¥第２部割込み編¥第２章ＵＳＡＲＴ割込み¥
にディレクトリの移動をしておいて下さい。

１．スタートアップを変更する。

　ＵＳＡＲＴの送受信割り込みを有効にする準備をします。

１）プログラムリスト

file　“StartupD.asm”

```
;/*****************************************************************************/
;/*      <MOD>     StartupD.asm                                                 */
;/*      <役割>    スタートアップ (割込み対応 CAT204-KL5C8012)                  */
;/*                リンクの都合上、先頭に指定することが絶対条件                 */
;/*                $$Rx_init_value_は、knil（リンク）で作成されます。           */
;/*      <TAB>     ４タブ編集                                                   */
;/*****************************************************************************/
STACK       EQU     0xFFFF          ; ｽﾀｯｸﾎﾞﾄﾑ
BBR1        EQU     0x0             ; KL5C8012(MMU) BBR1
BR1         EQU     0x1             ;               BR1
BBR2        EQU     0x2             ;               BBR2
BR2         EQU     0x3             ;               BR2
BBR3        EQU     0x4             ;               BBR3
BR3         EQU     0x5             ;               BR3
BBR4        EQU     0x6             ;               BBR4
BR4         EQU     0x7             ;               BR4
```

```
17: ;/*****************************************************************************/
18: ;/*          スタートアップ                                                   */
19: ;/*****************************************************************************/
20:             CSEG
21: StartUp:
22:             ORG     0
23:             DI
24:             LD      SP,STACK
25:             IM      2
26:             LD      A,0
27:             LD      I,A
28: ;/*****************************************************************************/
29: ;/*          MMUの初期設定                                                    */
30: ;/*****************************************************************************/
31:             LD      A,LOW  $$R1_init_value_##
32:             OUT     (BBR1),A
33:             LD      A,HIGH $$R1_init_value_##
34:             OUT     (BR1),A
35:             LD      A,LOW  $$R2_init_value_##
36:             OUT     (BBR2),A
37:             LD      A,HIGH $$R2_init_value_##
38:             OUT     (BR2),A
39:             LD      A,LOW  $$R3_init_value_##
40:             OUT     (BBR3),A
41:             LD      A,HIGH $$R3_init_value_##
42:             OUT     (BR3),A
43:             LD      A,LOW  $$R4_init_value_##
44:             OUT     (BBR4),A
45:             JP      main_##
46: ;/*****************************************************************************/
47: ;/*          割込みベクター                                                   */
48: ;/*****************************************************************************/
49:             ORG     0x80
50:             DW      NON                 ; IR[ 0] P00
51:             DW      NON                 ; IR[ 1] P01
52:             DW      NON                 ; IR[ 2] P02
```

```
53:             DW      NON             ; IR[ 3] P03
54:             DW      NON             ; IR[ 4] P04
55:             DW      NON             ; IR[ 5] P05
56:             DW      NON             ; IR[ 6] P06(IRQ1)
57:             DW      NON             ; IR[ 7] P07(IRQ2)
58:             DW      TXRDY           ; IR[ 8] TXRDY
59:             DW      RXRDY           ; IR[ 9] RXRDY
60:             DW      NON             ; IR[10] TXEMPTY
61:             DW      NON             ; IR[11] TIMER A CH0
62:             DW      NON             ; IR[12] TIMER A CH1
63:             DW      NON             ; IR[13] TIMER B CH0 OUTS
64:             DW      NON             ; IR[14] TIMER B CH1 OUTS
65:             DW      TIMERB2         ; IR[15] TIMER B CH2 OUTS
66: ;/*************************************************************************/
67: ;/*          割込みハンドラー                                             */
68: ;/*************************************************************************/
69: NON::                               ; 割込み未使用
70:             EI
71:             RETI
72: TXRDY::                                 ; IR[ 8] TXRDY
73:             PUSH    AF
74:             PUSH    BC
75:             PUSH    DE
76:             PUSH    HL
77:             PUSH    IX
78:             PUSH    IY
79:             CALL    Txrdy_##
80:             POP     IY
81:             POP     IX
82:             POP     HL
83:             POP     DE
84:             POP     BC
85:             POP     AF
86:             EI
87:             RETI
88: RXRDY::                                     ; IR[ 9] RXRDY
```

```
 89:             PUSH    AF
 90:             PUSH    BC
 91:             PUSH    DE
 92:             PUSH    HL
 93:             PUSH    IX
 94:             PUSH    IY
 95:             CALL    Rxrdy_##
 96:             POP     IY
 97:             POP     IX
 98:             POP     HL
 99:             POP     DE
100:             POP     BC
101:             POP     AF
102:             EI
103:             RETI
104: TIMERB2::                              ; IR[15] TIMER B CH2 OUTS
105:             PUSH    AF
106:             PUSH    BC
107:             PUSH    DE
108:             PUSH    HL
109:             PUSH    IX
110:             PUSH    IY
111:             CALL    TimerB2_##
112:             POP     IY
113:             POP     IX
114:             POP     HL
115:             POP     DE
116:             POP     BC
117:             POP     AF
118:             EI
119:             RETI
120: ;/*****************************************************************************/
121: ;/*          終了（C ソースへのエントリー）                                   */
122: ;/*****************************************************************************/
123:             ORG     0x200
124:             END
```

［リストの説明］　前章に変更を加えた個所を説明します。

５８行：

ＴＸＲＤＹ割り込みのベクターテーブルです。

５９行：

ＲＸＲＤＹ割り込みのベクターテーブルです。

７２～１０３行：

ＴＸＲＤＹとＲＸＲＤＹ用の割り込みハンドラを追加しました。

Ｃ言語記述関数名は、ＴＸＲＤＹ＝“Ｔｘｒｄｙ（）”　ＲＸＲＤＹ＝“Ｒｘｒｄｙ（）”です。

２．ＵＳＡＲＴ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動作をさせます。

プログラム例に沿って説明します。

１）プログラムリスト

file “I_Usart.c”

```
/*****************************************************************************/
/*                                                                           */
/*  <サンプル>  割込み                                                       */
/*                                                                           */
/*  <MOD>       I_Usart.c                                                    */
/*  <役割>      USART(SIO) 関係                                              */
/*  <TAB>       ４タブ編集                                                   */
/*  <保守ﾂｰﾙ>   makefile 参照                                                */
/*  <使用ﾊｰﾄﾞ>  CAT-204-KL5C8012 エーワン(株)                                */
/*                                                                           */
/*****************************************************************************/
#include    <machine.H>
#include    "CAT204.H"
#include    "DemoCtl.h"
/*****************************************************************************/
/*      USART ボレート周期計算+ETC                                           */
/*                    Sys      = 7372800Hz                                   */
/*                    TIMERB   = 4分周                                       */
/*                    TXC/RXC = 1/16                                         */
/*****************************************************************************/
#define CLOCK       (7372800/4)         /* 調歩同期 ﾎﾞｰﾚｰﾄｼﾞｪﾈﾚｰﾄ計算       */
#define BRRN(b)     (CLOCK/(b*16))
#define NON         0                   /* ﾊﾟﾘﾃｨ    NON                  */
#define EVEN        1                   /*          EVEN                 */
#define ODD         2                   /*          ODD                  */
```

```
26: #define STX         0x2             /* 送信スタートコード                */
27: #define ETX         0x3             /* 送信ストップコード                */
28: /*****************************************************************************/
29: /*      変数宣言                                                             */
30: /*****************************************************************************/
31:         Uchar       K51Step;        /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                       */
32:         Uchar       K51Select;      /* ﾎﾞｰﾚｰﾄ選択                        */
33:
34:         Uchar       TxDat[16];      /* 送信ﾃﾞｰﾀ                          */
35:         Uchar       TxIdx;          /* 送信中 ｲﾝﾃﾞｯｸｽ                    */
36:         Uchar       TxCnt;          /* 送信残 ｶｳﾝﾀ                       */
37:         Uchar       RxDat[16];      /* 受信ﾃﾞｰﾀ                          */
38:         Uchar       RxIdx;          /* 受信中 ｲﾝﾃﾞｯｸｽ                    */
39:         Uchar       RxEnd;          /* 受信終了ﾌﾗｸﾞ                      */
40: const   Ushort      BpsTbl[] =
41: {
42:         2400,
43:         4800,
44:         9600,
45:         19200,
46:         38400,
47:         0,
48: };
49: /*****************************************************************************/
50: /*      Ｍｅｍ初期化                                                         */
51: /*****************************************************************************/
52: void    K51MemInitial(void)
53: {
54:     K51Step   = 0;                  /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                       */
55:     K51Select = 2;                  /* 9600BPS                           */
56: }
57: /*****************************************************************************/
58: /*      Ｉ／Ｏ初期化                                                         */
59: /*****************************************************************************/
60: void    K51IoInitial(void)
61: {
```

```
62:     /* 重要!! TXC/RXC にため SCR0 の"D0"を"0"にする */
63:     RsOpen(9600,8,0);
64: }
65: /*************************************************************************/
66: /*      RS232C  Open  bps    = ﾎﾞﾚｰﾄ[2400,4800,*9600,19200,38400]          */
67: /*                    ch     = ｷｬﾗｸﾀ[7,*8]                                 */
68: /*                    pty    = ﾊﾟﾘﾃｨ[*0=NON 1=EVEN 2=ODD]                  */
69: /*                    *      = ﾃﾞﾌｫﾙﾄ                                      */
70: /*                    固定   = x16 STOP=1bit                               */
71: /*                    ﾎﾞｰﾚｰﾄ = ﾀｲﾏ B ﾁｬﾈﾙ 1 の OUTBP を使用                  */
72: /*                    sys    = 7372800Hz                                   */
73: /*************************************************************************/
74: void    RsOpen(Ushort bps,Uchar ch,Uchar pty)
75: {
76:     Uchar   cyc;
77:     Uchar   mode;
78:
79:     cyc = BRRN(9600);                   /* ﾎﾞｰﾚｰﾄ ｼﾞｪﾈﾚｰﾄ計算 (ﾃﾞﾌｫﾙﾄｾｯﾄ)  */
80:     if (bps == 2400)        cyc = BRRN(2400);
81:     else if (bps == 4800)   cyc = BRRN(4800);
82:     else if (bps == 9600)   cyc = BRRN(9600);
83:     else if (bps == 19200)  cyc = BRRN(19200);
84:     else if (bps == 38400)  cyc = BRRN(38400);
85:
86:     outp(TCWDB1,0x0e);                  /* TIME-B1 outL+PWM+sys/4       */
87:     outp(TCNTB1,cyc-1);                 /*         周期設定             */
88:     outp(TCNTB1,(cyc/2)-1);             /*         巾設定(ﾃﾞｭｰﾃｨ 50%)    */
89:
90:     outp(K51COM0,0);                    /* KP51  ﾀﾞﾐｰ                   */
91:     outp(K51COM0,0);                    /*       ﾀﾞﾐｰ                   */
92:     outp(K51COM0,0);                    /*       ﾀﾞﾐｰ                   */
93:     outp(K51COM0,0x40);                 /*       ｿﾌﾄﾘｾｯﾄ                */
94:     mode = 0x4e;                        /* Mode  Stop=1 NON 8bit x16(ﾃﾞﾌｫﾙﾄ)*/
95:     if (pty == EVEN)      mode |= 0x30;     /*  ﾊﾟﾘﾃｨ=EVEN              */
96:     else if (pty == ODD)  mode |= 0x20;     /*  ﾊﾟﾘﾃｨ=ODD               */
97:     if (ch == 7)          mode &= ~(0x4);   /*  ｷｬﾗｸﾀ 7                 */
```

```
 98:     outp(K51COM0,mode);                 /*      設定                    */
 99:     outp(K51COM0,0x36);                 /* Cmd   RTS ERR RX=EI DTR TX=DI   */
100: }
101: /*************************************************************************/
102: /*      TxStart     RS232C 文字列送信(TXRDY 割込み準備)                    */
103: /*************************************************************************/
104: void    TxStart(Uchar *tx,Uchar cnt)
105: {
106:     TxDat[0] = STX;                     /* スタートコード                  */
107:     memcpy(&TxDat[1],tx,cnt);           /* 送信ﾊﾞｯﾌｧｰに記憶                */
108:     TxDat[cnt+1] = ETX;                 /* ストップコード                  */
109:     TxIdx = 0;                          /* 送信中ｲﾝﾃﾞｯｸｽ                   */
110:     TxCnt = cnt+2;                      /* 送信残ｶｳﾝﾀ                     */
111:     outp(K51COM0,0x37);                 /* Cmd   RTS ERR RX=EI DTR TX=EI   */
112: }
113: /*************************************************************************/
114: /*      Txrdy       TXRDY 割込み      割り込みハンドラより呼ばれる          */
115: /*************************************************************************/
116: void    Txrdy()
117: {
118:     if (TxCnt == 0) outp(K51COM0,0x36); /* 送信終了 TX=DI にする           */
119:     else {
120:         outp(K51DAT0,TxDat[TxIdx]);
121:         ++TxIdx;
122:         --TxCnt;
123:     }
124: }
125: /*************************************************************************/
126: /*      Rxrdy       RXRDY 割込み      割り込みハンドラより呼ばれる          */
127: /*************************************************************************/
128: void    Rxrdy()
129: {
130:     Uchar  dt;
131:
132:     dt = inp(K51DAT0);                  /* 受信                    */
133:     if ((inp(K51COM0) & 0x38)) {        /* USART Error?            */
```

```
134:        outp(K51COM0,0x32);                 /* ERR Reset RX=DI TX=DI    */
135:        RxIdx = 0;
136:    }
137:    else {                                  /* 正常受信                 */
138:        if (dt == STX) RxIdx = 0;           /* スタートコード           */
139:        else if (dt == ETX) RxEnd = ON;     /* 受信終了ﾌﾗｸﾞｾｯﾄ          */
140:        else {                              /* 受信データ               */
141:            RxDat[RxIdx++] = dt;
142:            if (RxIdx >= sizeof(RxDat)) RxIdx = 0;  /* ﾊﾞﾌｧｰｵｰﾊﾞｰの防止    */
143:        }
144:    }
145: }
146: /*************************************************************************/
147: /*      UsartDemo()     ＵＳＡＲＴデモ                                   */
148: /*************************************************************************/
149: void    K51Demo()
150: {
151:     K51Sequence();                  /* Usart 操作ｺﾝﾄﾛｰﾙ                 */
152:     switch(K51Step) {
153:     case 0:
154:         break;
155:     case 1:
156:         RsOpen(BpsTbl[K51Select],8,0);  /* RS232C Open                  */
157:         K51Step++;
158:         break;
159:     case 2:
160:         RxEnd = OFF;                    /* 受信終了ﾌﾗｸﾞ                 */
161:         TxStart("I t r u t C",11);      /* 送信/受信割込み開始          */
162:         TmStart(TM1,500);               /* ﾀｲﾑｵｰﾊﾞｰ管理                 */
163:         K51Step++;
164:         break;
165:     case 3:
166:         if (RxEnd == ON) {              /* 受信終了を待つ               */
167:             RxDat[RxIdx] = 0;           /* 確認用の表示データ作成       */
168:             GotoxyMemSet(0,0,RxDat);    /* 受信割込みﾃﾞｰﾀ表示           */
169:             K51Step = 4;
```

```
170:        }
171:        else if (TmUpTest(TM1) == ON) {     /* ﾀｲﾑｵｰﾊﾞｰ                    */
172:            GotoxyMemSet(0,0,"ﾀｲﾑｵｰﾊﾞｰ ");  /* Error 表示                  */
173:            K51Step = 4;
174:        }
175:        break;
176:    case 4:
177:        RxEnd = OFF;                    /* 受信終了ﾌﾗｸﾞ                 */
178:        TxStart(" n e r p S ",11);       /* 送信/受信割込み開始          */
179:        TmStart(TM1,500);               /* ﾀｲﾑｵｰﾊﾞｰ管理                 */
180:        K51Step++;
181:        break;
182:    case 5:
183:        if (RxEnd == ON) {              /* 受信終了を待つ               */
184:            RxDat[RxIdx] = 0;           /* 確認用の表示データ作成       */
185:            GotoxyMemSet(0,0,RxDat);    /* 受信割込みﾃﾞｰﾀ表示           */
186:            K51Step = 2;
187:        }
188:        else if (TmUpTest(TM1) == ON) {     /* ﾀｲﾑｵｰﾊﾞｰ                    */
189:            GotoxyMemSet(0,0,"ﾀｲﾑｵｰﾊﾞｰ ");  /* Error 表示                  */
190:            K51Step = 2;
191:        }
192:        break;
193:    }
194: }
195: /*****************************************************************************/
196: /*      K51Sequence()   Usart 操作ｺﾝﾄﾛｰﾙ                                     */
197: /*****************************************************************************/
198: void    K51Sequence()
199: {
200:    Uchar   port;
201:    Uchar   dec[5+1];
202:
203:    port = GetUpPort(1);                /* PB[P30]->PB[P33]             */
204:    if (port & 0xe) {                   /* PB[P31]->PB[P33] ON(立上) ?  */
205:        if (port & 0x2) {               /* PB[P31] ON ? 送信ｽﾀｰﾄ/ｽﾄｯﾌﾟ  */
```

```
206:             if (K51Step == 0) K51Step = 1;
207:             else              K51Step = 0;
208:         }
209:         if (K51Step == 0) {             /* 停止中のみ受け付ける             */
210:             if (port & 0x4) {           /* PB[P32] ON ? ﾎﾞﾚｰﾄ下げ?          */
211:                 if (K51Select != 0) --K51Select;
212:             }
213:             else if (port & 0x8) {      /* PB[P33] ON ? ﾎﾞﾚｰﾄ上げ?          */
214:                 if (BpsTbl[K51Select+1] != 0) ++K51Select;
215:             }
216:             Bin2AdecN(dec,BpsTbl[K51Select],5); /* 表示用データ作成         */
217:             GotoxyMemSet(0,1,dec);
218:         }
219:     }
220: }
```

［リストの説明］　前章に変更を加えた個所を説明します。

２６～２７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。

３４～３６行：

送信割り込み関数で使用する変数の宣言です。

３７～３９行：

受信割り込み関数で使用する変数の宣言です。

９９行：

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。

１０４～１１２行：

送受信割り込みを開始させる関数です。

１０６～１１０行は、文字列送信を割り込みで処理するための準備です。

準備終了後、送信をイネーブル状態にし“ＴＸＲＤＹ”割り込みで文字列データを１バイトごとに割り込み発生毎に送信をします。

１１６～１２４行：

ＴＸＲＤＹ（送信）割り込みハンドラより呼ばれる関数です。

１バイト毎に送信可能状態（ＴＸＲＤＹ）になると割り込みが発生します。

送信するデータがなくなった場合は、送信をディセーブル状態にして割り込みを止めます。

１２８～１４５行：

ＲＸＲＤＹ（受信）割り込みハンドラより呼ばれる関数です。

１バイト毎に受信完了後、データ呼びだし可能状態（ＲＸＲＤＹ）になると割り込みが発生します。

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。

エラーが発生した場合は、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。

１４９～１９４行：

ＵＳＡＲＴ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［Ｐ３１］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。

３．メインコントロール部の割り込みコントローラを変更する。

ＵＳＡＲＴ割り込みを可能にするため、割り込みコントローラの設定変更が必要です。
プログラム例に沿って説明します。

１）プログラムリスト

file “Cat204i2.c”

```
  1: /*****************************************************************************/
  2: /*                                                                           */
  3: /*  ＜サンプル＞ 割込み                                                      */
  4: /*                                                                           */
  5: /*  ＜MOD＞      Cat204i2.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-204-KL5C8012 エーワン(株)                               */
 10: /*                                                                           */
 11: /*****************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.H"
 14: #include    "DemoCtl.h"
 15: /*****************************************************************************/
 16: /*      外部変数使用                                                         */
 17: /*****************************************************************************/
 18: extern  short      BuzzerHz;          /* TIMER  Buzzer Hz                  */
 19: extern  Uchar      K51Step;           /* USART  ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                */
 20: extern  Uchar      K51Select;         /*        ﾎﾞｰﾚｰﾄ選択                 */
 21: extern  Uchar      MelStep;           /* Melody ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                */
 22: extern  Uchar      MelMode;           /*        演奏ﾓｰﾄﾞ                    */
 23: extern  Uchar      MelSelect;         /*        自動選曲                    */
 24: /*****************************************************************************/
 25: /*      変数宣言                                                             */
 26: /*****************************************************************************/
 27:         Uchar      ModeStep;          /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                */
 28:         Uchar      Shift;             /* shift ﾊﾟﾀｰﾝ                       */
```

```
29: /*****************************************************************************/
30: /*      main()                                                               */
31: /*****************************************************************************/
32: void    main(void)
33: {
34:     outp(SCR1,0xc0);                /* SYS    ExtMem 0wait ExtIO 1wait  */
35:     outp(SCR0,0x10);                /*        (OUTBx,OUTAx)             */
36:     SoftWait1ms(20);                /* Power On Wait(20ms) 安定待ち     */
37:     MemInitial();                   /* ﾒﾓﾘ系初期化                       */
38:     IoInitial();                    /* I/O 系初期化                      */
39:     TmStart(TM0,20);                /* ﾁｬﾀﾘﾝｸﾞ防止ｽﾀｰﾄ                   */
40:     ei();                           /* <- 割り込み許可                   */
41:     while(1) {
42:         if (TmUpTest(TM0) == ON) {
43:             TmStart(TM0,20);        /* ﾁｬﾀﾘﾝｸﾞ防止ｽﾀｰﾄ                   */
44:             SigInput();             /* Signal Input  Process             */
45:             ModeCntrol();           /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                        */
46:             SigOutput();            /* Signal Output Process             */
47:         }
48:         AllLcdDisp();               /* LCD    全画面表示                 */
49:     }
50: }
51: /*****************************************************************************/
52: /*      Ｍｅｍ初期化                                                         */
53: /*****************************************************************************/
54: void    MemInitial(void)
55: {
56:     ModeStep = 0;                   /* ﾓｰﾄﾞｺﾝﾄﾛｰﾙ用ｽﾃｯﾌﾟ                 */
57:     Shift = 0;                      /* Led Disp Patan Initial            */
58:
59:     PioMemInitial();                /* PIO    Mem 初期化                 */
60:     LcdMemInitial();                /* LCD    Mem 初期化                 */
61:     TimMemInitial();                /* TIM    Mem 初期化                 */
62:     K51MemInitial();                /* SIO    Mem 初期化                 */
63:     MelMemInitial();                /* Melody Mem 初期化                 */
64: }
```

```
65: /*****************************************************************************/
66: /*        Ｉ／Ｏ初期化                                                       */
67: /*****************************************************************************/
68: void    IoInitial(void)
69: {
70:     PioIoInitial();                  /* PIO    I/O 初期化                 */
71:     LcdIoInitial();                  /* LCD    I/O 初期化                 */
72:     TimIoInitial();                  /* TIM    I/O 初期化                 */
73:     K51IoInitial();                  /* SIO    I/O 初期化                 */
74:     MelIoInitial();                  /* Melody I/O 初期化                 */
75:
76:     outp(LERL,0);                    /* 割込<- ﾚﾍﾞﾙ/ｴｯｼﾞ設定              */
77:     outp(LERH,0x80);                 /*      <- IR[15]TIMER B CH2 OUTS ｴｯｼﾞ*/
78:     outp(IVR,0x80);                  /*      <- 先頭ﾍﾞｸﾀ 80H               */
79:     outp(PGRL,0);                    /*      <- ﾌﾟﾗｲｵﾘﾃｨの設定             */
80:     outp(PGRH,0x2);                  /*      <- IR[ 9]RXRDY(High)         */
81:     outp(IMRL,0xFF);                 /*      <- ﾏｽｸﾚｼﾞｽﾀ                   */
82:     outp(IMRH,0x7C);                 /*      <- IR[15]TIMER B CH2 OUTS    */
83:                                      /*      <- IR[ 9]RXRDY               */
84:                                      /*      <- IR[ 8]TXRDY               */
85: }
86: /*****************************************************************************/
87: /*      ModeCntrol()    ﾓｰﾄﾞｺﾝﾄﾛｰﾙ                                           */
88: /*****************************************************************************/
89: void    ModeCntrol()
90: {
91:     if (GetUpPort(1) & 0x1) {        /* PB[P30] ON?(立上)                 */
92:         if (ModeStep < 10)      ModeStep = 10;      /*  PIO   Goto TEST  */
93:         else if (ModeStep < 20) ModeStep = 20;      /*  Timer Goto TEST  */
94:         else if (ModeStep < 30) ModeStep = 30;      /*  USART Goto TEST  */
95:         else if (ModeStep < 40) ModeStep = 40;      /*  Demo  Melody     */
96:         else                    ModeStep = 0;       /*  ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ    */
97:         Buzzer(0);                                  /*  強制 OFF          */
98:     }
99:     switch(ModeStep) {
100:    case 0:
```

```
101:        GotoxyMemSet(0,0,"CAT204&BF3000 by");       /* ｵｰﾌﾟﾆﾝｸﾞﾒｯｾｰｼﾞ        */
102:        GotoxyMemSet(0,1,"Interrupt  [P30]");
103:        ModeStep++;
104:        break;
105:    case 1:
106:        RunRun();
107:        break;
108:    case 10:                                        /*  PIO   TEST          */
109:        GotoxyMemSet(0,0,"PIO              ");
110:        GotoxyMemSet(0,1,"SW[P47]->SW[P40]");
111:        ModeStep++;
112:        break;
113:    case 11:
114:        PioDemo();
115:        break;
116:    case 20:                                        /*  Timer TEST          */
117:        GotoxyMemSet(0,0,"Timer/Counter   ");
118:        GotoxyMemSet(0,1,"0000Hz[+P33-P32]");
119:        BuzzerHz = 0;                               /* Buzzer Hz            */
120:        ModeStep++;
121:        break;
122:    case 21:
123:        TimerDemo();
124:        RunRun();
125:        break;
126:    case 30:                                        /*  USART TEST          */
127:        GotoxyMemSet(0,0,"Usart 8,n,1[P31]");
128:        GotoxyMemSet(0,1,"09600b[+P33-P32]");
129:        K51Step   = 0;                              /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ          */
130:        K51Select = 2;                              /* 9600BPS              */
131:        ModeStep++;
132:        break;
133:    case 31:
134:        K51Demo();
135:        RunRun();
136:        break;
```

```
137:      case 40:                                          /*  Demo  Melody        */
138:          GotoxyMemSet(0,0,"Melody          ");
139:          GotoxyMemSet(0,1,"P31[M]33[ｽ]32[ｾ]");
140:          MelStep   = 0;                                /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ          */
141:          MelMode   = 0;                                /* 演奏ﾓｰﾄﾞ             */
142:          MelSelect = 0;                                /* 自動選曲             */
143:          ModeStep++;
144:          break;
145:      case 41:
146:          Melody();
147:          RunRun();
148:          break;
149:      }
150: }
151: /*************************************************************************/
152: /*      RunRun()        CPU 走行表示                                     */
153: /*************************************************************************/
154: void    RunRun()
155: {
156:      if ((Shift <<= 1) == 0) Shift = 1;          /* LED Shift 表示       */
157:      PutOutPort(Shift,'=');
158: }
159: /*************************************************************************/
160: /*      SoftWait1ms()   1ms 単位 ｿﾌﾄﾀｲﾏｰ                                 */
161: /*************************************************************************/
162: void    SoftWait1ms(Ushort ms)
163: {
164:      while(ms-- != 0) {
165:          Wait1ms();
166:      }
167: }
168: /*************************************************************************/
169: /*      Wait1ms()       1ms ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                */
170: /*************************************************************************/
171: void    Wait1ms()
172: {
```

```
173:     _asm_("\n          PUSH    HL              \n");
174:     _asm_("\n          LD      HL,1228         \n");   /* 1228*6=7372cyc   */
175:     _asm_("\n W01:                             \n");
176:     _asm_("\n          DEC     HL              \n");   /* cyc = 1          */
177:     _asm_("\n          LD      A,L             \n");   /*     = 1          */
178:     _asm_("\n          OR      H               \n");   /*     = 1          */
179:     _asm_("\n          JP      NZ,W01          \n");   /*     = 3          */
180:     _asm_("\n          POP     HL              \n");   /*    += 6          */
181: }
182: /*****************************************************************************/
183: /*      SoftWait10us()  10us 単位 ｿﾌﾄﾀｲﾏｰ                                    */
184: /*****************************************************************************/
185: void    SoftWait10us(Ushort us)
186: {
187:     while(us-- != 0) {
188:         Wait10us();
189:     }
190: }
191: /*****************************************************************************/
192: /*      Wait10us()      10us ｿﾌﾄﾀｲﾏｰ (7.3728MHz)  Non Wait                   */
193: /*****************************************************************************/
194: void    Wait10us()
195: {
196:     _asm_("\n          PUSH    HL              \n");
197:     _asm_("\n          LD      HL,13           \n");   /* 13*6=78cyc       */
198:     _asm_("\n W02:                             \n");
199:     _asm_("\n          DEC     HL              \n");   /* cyc = 1          */
200:     _asm_("\n          LD      A,L             \n");   /*     = 1          */
201:     _asm_("\n          OR      H               \n");   /*     = 1          */
202:     _asm_("\n          JP      NZ,W02          \n");   /*     = 3          */
203:     _asm_("\n          POP     HL              \n");   /*    += 6          */
204: }
```

［リストの説明］　　前章に変更を加えた個所を説明します。

このモジュールでの変更は、割り込みコントローラへの設定変更のみです。

８０行：

割り込みプライオリティの変更をしました。

ＩＲ［9］ＲＸＲＤＹ　を“ＨＩＧＨ”グループにしました。

通常ＲＸＲＤＹは、いつ発生するか不定のため“ＨＩＧＨ”グループにした方が良いと思います。

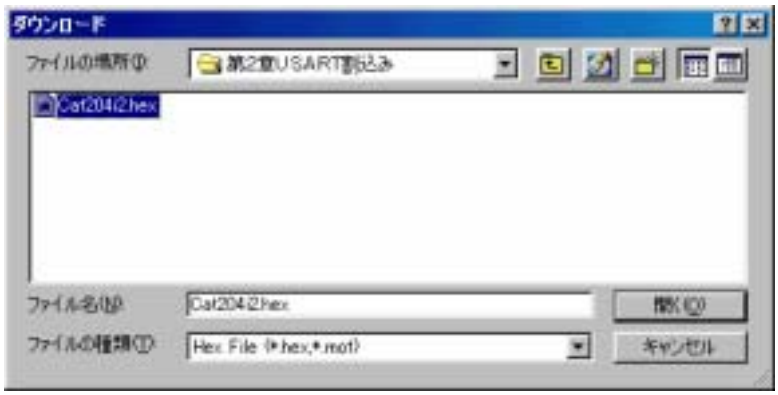
８２行：

マスク解除を、ＩＲ［１５］ＴＩＭＥＲ　Ｂ　ＣＨ２　ＯＵＴＳ、ＩＲ［9］ＲＸＲＤＹ，ＩＲ［8］ＴＸＲＤＹの３つ解除しました。

以上の変更で、タイマー／ＵＳＡＲＴ割り込み対応のシステムに変貌したわけです。

動作的には、ＵＳＡＲＴが文字列通信部分を変更しただけですが、動作確認してみましょう。

ＡＢＣｗｉｎでダウンロード後、プログラム実行をして下さい。



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　“I_Melody.c”

```
  1: /*************************************************************************/
  2: /*                                                                       */
  3: /*  ＜サンプル＞  割込み                                                 */
  4: /*                                                                       */
  5: /*  ＜MOD＞       I_Melody.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-204-KL5C8012 エーワン(株)                          */
 10: /*                                                                       */
 11: /*************************************************************************/
 12: #include    <machine.H>
 13: #include    "CAT204.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                                                             */
/*****************************************************************************/
        Ushort      MelodyHz;       /* Melody Hz                        */
        Uchar       MelMode;        /* 演奏ﾓｰﾄﾞ                         */
        Uchar       MelSelect;      /* 自動選曲                         */
        Uchar       MelStep;        /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
        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 */
        0
```

```
55: };
56: const   Ushort      MusicTbl[3][SCMAX] =    /* 自動演奏の楽譜（最大32ﾏﾃﾞ）   */
57: {                                                          /*  ﾈｺﾌﾝｼﾞｬｯﾀ   */
58:     { ra, so, do, Do, Do, ra, so, do, Do, Do,
59:       ra, so, do, Do, rA, Do, s0, si, si},
60:                                                           /*  ｲﾇﾉｵﾏﾜﾘｻﾝ   */
61:     { mi, do, do, do, mi, do, do, do,fha,fha, mi, mi, re,
62:      fha,fha, mi, mi, re, re, ra, ra, so,fha, mi, re, do},
63:                                                           /*  ｱﾏﾘﾘｽ       */
64:     { so, ra, so, Do, so, ra, so, ra, ra, so, ra, so,fha, mi, re, mi, do, 0},
65: };
66: const   Ushort      RhythmTbl[3][SCMAX] =   /* 自動演奏のリズム(最大32ﾏﾃﾞ）  */
67: {                                                          /*  ﾈｺﾌﾝｼﾞｬｯﾀ   */
68:     {250,250,500,500,500,250,250,500,500,500,
69:      250,250,500,500,500,500,500,500,500},
70:                                                           /*  ｲﾇﾉｵﾏﾜﾘｻﾝ   */
71:     {250,250,250,250,250,250,250,250,250,250,250,250,1000,
72:      250,250,250,250,250,250,250,250,250,250,250,250,1000,},
73:                                                           /*  ｱﾏﾘﾘｽ       */
74:     {500,500,500,500,500,500,1000,500,500,500,500,250,250,250,250,500,500, 0},
75: };
76: const   Uchar       *MelodyTblA[] =     /*  自動演奏の曲名                 */
77: {
78:         "          ",
79:         "ﾈｺﾌﾝｼﾞｬｯﾀ",
80:         "ｲﾇﾉｵﾏﾜﾘｻﾝ",
81:         "ｱﾏﾘﾘｽ    ",
82: };
83: /*************************************************************************/
84: /*      Melody          ﾒﾛﾃﾞｨｰｺﾝﾄﾛｰﾙ                                     */
85: /*************************************************************************/
86: void    Melody()
87: {
88:     MelodySequence();                   /* ﾒﾛﾃﾞｨｰ操作ｺﾝﾄﾛｰﾙ               */
89:     if (MelMode == 0) ManualMelody();   /* 手動演奏                       */
90:     else              AutoMelody();     /* 自動演奏                       */
```

```
 91: }
 92: /*************************************************************************/
 93: /*      MelodySequence()    ﾒﾛﾃﾞｨｰ操作ｺﾝﾄﾛｰﾙ                                 */
 94: /*************************************************************************/
 95: void    MelodySequence()
 96: {
 97:     Uchar   port;
 98:
 99:     port = GetUpPort(1);                /* PB[P30]->PB[P33]                */
100:     if (port & 0xe) {                   /* PB[P31]->PB[P33] ON(立上) ?      */
101:         Buzzer(0);                      /* Buzer OFF                       */
102:         if (port & 0x2) {               /* PB[P31] ON ? ﾓｰﾄﾞ変更           */
103:             MelStep = 0;                /* 強制停止                        */
104:             if (MelMode == 0) {         /* 現在手動 ?                      */
105:                 MelMode   = 1;          /* 自動に変更                      */
106:                 MelSelect = 1;          /* 自動選曲                        */
107:             }
108:             else {                      /* 現在自動 ?                      */
109:                 MelMode   = 0;          /* 手動に変更                      */
110:                 MelSelect = 0;          /* 自動選曲                        */
111:             }
112:         }
113:         if (MelMode != 0) {             /* 自動 ?                          */
114:             if (port & 0x4) {           /* PB[P32] ON ?  選曲              */
115:                 MelStep = 0;            /* 強制停止                        */
116:                 if (++MelSelect > 3) MelSelect = 1;
117:             }
118:             if (port & 0x8) {           /* PB[P33] ON ? 自動演奏開始       */
119:                 if (MelStep == 0) MelStep = 1;      /*  開始               */
120:                 else              MelStep = 0;      /*  停止               */
121:                 Music   = 0;
122:             }
123:         }
124:         if (MelMode == 0) GotoxyMemSet(4,1,"M");    /* 手動表示            */
125:         else              GotoxyMemSet(4,1,"A");    /* 自動表示            */
126:         GotoxyMemSet(7,0,(Uchar *)MelodyTblA[MelSelect]);   /* 曲名表示    */
```

```
127:     }
128: }
129: /*****************************************************************************/
130: /*       Ｍｅｍ初期化                                                        */
131: /*****************************************************************************/
132: void    MelMemInitial(void)
133: {
134:     MelodyHz  = 0;                  /* Melody Min Hz                    */
135:     MelMode   = 0;                  /* 演奏ﾓｰﾄﾞ                         */
136:     MelStep   = 0;                  /* ｺﾝﾄﾛｰﾙｽﾃｯﾌﾟ                      */
137:     MelSelect = 0;                  /* 自動選曲                         */
138: }
139: /*****************************************************************************/
140: /*       Ｉ／Ｏ初期化                                                        */
141: /*****************************************************************************/
142: void    MelIoInitial(void)
143: {
144:     Buzzer(0);                      /* Buzer OFF                        */
145: }
146: /*****************************************************************************/
147: /*       ManualMelody   手動ﾒﾛﾃﾞｨｰ演奏                                      */
148: /*****************************************************************************/
149: void    ManualMelody()
150: {
151:     Uchar   port;
152:
153:     switch(MelStep) {
154:     case 0:
155:         _strcpyW(Doremi,(Ushort *)DoremiTbl);   /* ﾄﾞﾚﾐ音階周波数(初期準備)*/
156:         MelStep++;
157:         break;
158:     case 1:
159:         if (GetInPort(0) & 0xff) {      /* P40->P47 どれかがＯＮしたか？    */
160:             port = GetUpPort(0);
161:             if (port & 0x1) {           /* SW[P40] 立上がり ON (ド)         */
162:                 Buzzer(MelodyHz = Doremi[7]);
```

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

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

［リストの説明］　前章に変更を加えた個所を説明します。

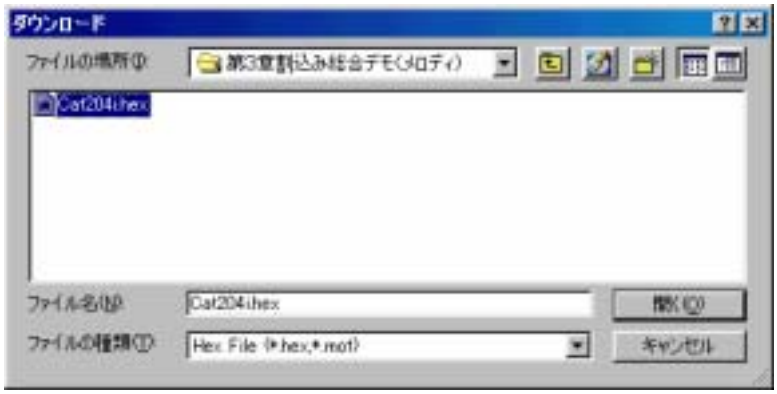
２０９行：　２１５行：　２２０行：　２２６行：

自動演奏のソフトタイマ使用部分を、割り込み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。

ＡＢＣｗｉｎ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、このシステムには改造する部分（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　3月　著者