目次

１．H8SPAS について

H8SPAS はｈ８ｓ/２０００シリーズ用 PASCAL コンパイラです。ROM 化を意識したオプティマイズされたコード生成を行います。また割り込み機能にも対応しているので、プログラム作成も簡単です。

特徴は以下のとおりです。

- INTERRUPT 宣言により、割り込みルーチンのプログラムが可能。
- EXTERNAL 宣言により、外部ファイルにあるルーチンを呼び出すことが可能。
- FORWARD　宣言により前方参照を可能にします。また外部に対しては PUBLIC の宣言と同様な効果があります。（そのため PUBLIC は定義されていない）
- ｛$I xxxx.pas｝と宣言することにより、ほかのパスカルファイルのインクルードが可能になります。
- ｛$L xxxx.asm｝と宣言することにより、アセンブラルーチンのインクルードが可能になります。
- ３２ビット型の整数　LONGINT 型の使用が可能。
- ４８ビット型の実数　REAL 型の使用可能。　（オプションです）
- ポインタ型のデータ使用可能。
- レコード型のデータ使用可能。複雑な型のデータもレコード型にすることによりプログラムの間違いも少なく、簡単にプログラムできます。
- マルチタスクオペレーティングシステム　MTOS をインクルードすると、マルチタスクプログラムが簡単に実現することができます。

２．基本的な言語要素

２．１　予約語

ABSOLUTE	AND	ARRAY	BEGIN
CASE	CONST	DIV	DO
DOWNTO	ELSE	END	EXTERNAL
FILE	FOR	FORWARD	FUNCTION
IF	IN	INTERFACE	INTERRUPT
IMPLEMENTATION		MOD	NEXTOF
NIL	NOT	OF	OR
PROCEDURE	PROGRAM	RECORD	REPEAT
SHL	SHR	TO	THEN
TYPE	VAR	XOR	UNIT
UNTIL	USES	WHILE	WITH

２．２　標準関数（手続き）

ABS	ASSIGN	BLOCKREAD	BLOCKWRITE
CHDIR	CHR	CLOSE	COPY
COS	DISPOSE	DELETE	ERASE
EOF	EXIT	FILESIZE	FINDFIRST
FINDNEXT	LENGTH	MAXAVAIL	MEMAVAIL
MKDIR	NEW	ODD	ORD
POS	PRED	READ	READKEY
READLN	RESET	REWRITE	RMDIR
SIN	SIZEOF	SQRT	SUCC
STR	TRUNC	UPCASE	VAL
WRITE	WRITELN		

２．３　サポートされている型

- BOOLEAN　　TRUE、FALSE の二値を持つ型。　１BYTE の大きさ。
- BYTE　　0..$ff　までの値をとる型。　　１BYTE の大きさ。
- CHAR　　文字を表す型。　　１BYTE の大きさ。
- INTEGER　　－３２７６８から　＋３２７６７までの値をとる型。２BYTE。
- LONGINT　　３２BIT のサイズを持つ。整数型。
- スカラー型
- 部分範囲型
- 文字列型　　文字列を表す型。１から２５５までの長さの文字列を表現できる。
- レコード型
- ポインター型

２．４　あらかじめ準備されている定数

あらかじめ準備されている定数には次のものがあります。

- TRUE　　BOOLEAN 用の値。
- FALSE　　BOOLEAN 用の値。
- PI　　円周率　（実数型）
- DATE　　コンパイルされた日時を表す文字列。これを使い、製品の版数管理等が可能です。"compiled: 2005/05/04 21:53:31" という形式です。

２．５　コメント

コメントは３種類サポートされています。

- { は次の } が来るまでがコメントとなります。
- （* は次の *）が来るまでがコメントとなります。
- // は//から行の終わりまでがコメントとなります。

２．６　コンパイラ指令

以下のコンパイラ指令があります。

- {$M aaa,bbb,ccc }　または　{$MEM aaa,bbb,ccc }　aaa はヒープの先頭アドレス、bbb はスタックのアドレス、ccc はプログラムの先頭アドレスです。これらはカンマのみで省略可能です。　例　｛＄M　＄１０００,, ＄０　｝
- {$C-} {$C+}　コメント生成を禁止　＄C-　または許可　＄C+　します。
- {$V-} {$V+}　VER で文字列渡しの場合の厳密チェック　＄V+　非チェック　＄V-
- {$I-} {$I+}　IO チェックを行う　＄I+　行わない　＄I-
- {$I xxx.pas }　ｘｘｘ. PAS をインクルードします。
- {$L xxx.asm }　ｘｘｘ. ASM をインクルードします。
- {$DEF xxx }　ｘｘｘを定義します。＄IFDEF で判定します。
- {$IFDEF aaaa }...{$ENDIF }　aaa が＄DEF で定義されていたら、＄ENDIF までの文をコンパイルします。

３．割り込みベクタ

割り込みベクタはアセンブラで記述する必要があります。この内容はワンパターンで、下に示したようにアドレスの定義を行います。

- 電源投入時のプログラム開始アドレスを定義
- 割り込みの種類ごとに実行ルーチンアドレスを定義

割り込みベクタの例を下に示します。Jtbl+4*xx 以外のところが、各ルーチンのスターとアドレスになります。Jtbl+4*xx　は無効な割り込み等があった場合のトラップ用等のために別の場所に飛ばしてから、トラップ、アドレスの検出等を行うためのものです。あまり気にする必要はありません。

```
;
;====================================
;===                                ===
;===  vecter    ( vect26.asm )      ===
;===   aug. 6 '96  by @             ===
;===                                ===
;====================================
;*
;* aug.  6 '96  for h8s/2600
;*
vctaddr equ     0
;
        org     vctaddr
¥vect:  dl      ¥powon          ; power on
        dl      ¥powon          ; reset
        dl      jtbl+4*02       ; reserved
        dl      jtbl+4*03       ; reserved
        dl      jtbl+4*04       ; reserved
        dl      jtbl+4*05       ; trace
        dl      jtbl+4*06       ; reserved
           :
        途中省略
           :
        dl      jtbl+4*84       ; eri1
        dl      jtbl+4*85       ; rxi1
        dl      jtbl+4*86       ; txi1
        dl      jtbl+4*87       ; tei1
        dl      jtbl+4*88       ; eri2
        dl      jtbl+4*89       ; rxi2
        dl      jtbl+4*90       ; txi2
        dl      jtbl+4*91       ; tei2
;
```

```
;================================
;==                              ==
;==   jump table                 ==
;==                              ==
;================================
;
jtbl:   bsr     goapint         ; power on
        bsr     goapint         ; reset
        bsr     goapint         ; reserved
        bsr     goapint         ; reserved
        bsr     goapint         ; reserved
        bsr     goapint         ; trace
        bsr     goapint         ; reserved
        bsr     goapint         ; nmi
           :
        途中省略
           :
        bsr     goapint         ; eri2
        bsr     goapint         ; rxi2
        bsr     goapint         ; txi2
        bsr     goapint         ; tei2
;
;================================
;==                              ==
;==  go apprication              ==
;==                              ==
;================================
;
goapint:bra     goapint         ; halt
;
;================================
;==                              ==
;==   power on entry             ==
;==                              ==
;================================
;
indhp   equ     h'ffec00        ; sysram area
¥powon: mov.l   program-4,r0
        mov.l   r0,indhp        ; heap
        jmp     program
;
        end
```

４．ランタイムルーチン

５．コンパイルの手順（プログラム完成までの手順）

６．エラーメッセージ一覧

001: ';' が必要です
002: ':' が必要です
003: ',' が必要です
004: '(' が必要です
005: ')' が必要です
006: '=' が必要です
007: ':=' が必要です
008: '[' が必要です
009: ']' が必要です
010: '.' が必要です
011: '..' が必要です
012: BEGIN が必要です
013: DO が必要です
014: END or ; が必要です
015: OF が必要です
016: UNTIL が必要です
017: THEN が必要です
018: TO or DOWNTO が必要です
019: CASE selector が必要です
020: Boolean 式が必要です
021: Absolute 変数が必要です
022: Integer 定数が必要です
023: Integer 式が必要です
024: Integer 変数が必要です
025: Integer or real 定数が必要です
026: Integer or real 式が必要です
027: Integer or real 変数が必要です
028: Pointer 変数が必要です
029: Record 変数が必要です
030: Simple 型が必要です
031: Simple 式が必要です
032: String 定数 not allowed
033: String 式が必要です
034: String 変数が必要です
035: Typed const は許されていません
036: Type identifier が必要です
037: 前の定義と一致していません
041: 未定義の identifier です
042: 未定義の pointer 型です
043: identifier の二重定義です
044: 型があっていません
045: 定数が範囲を超えています

046: TRY はネストできません
047: 型があっていません
048: 結果の型が未定義です
049: 文字列の長さが未定義です
050: 無効な変数への参照です
051: Case tags がぶつかっています
052: 上限と下限が逆転してます
053: 予約語です
054: ０で割っています
055: 文字列が１行を超えています
058: 無効な文字があります
059: Forward proc/func 本体がありません
060: Forward は入れ子に出来ません
061: 自分自身への参照をしています
062: 奇数アドレスに割り当てしています
063: 変数名か型名が必要です
064: 無効なコンパイラ指令です
065: サポートされていない CPU です
069: fields の順番が間違っています
070: uses　は使えません
071: implementation　が必要です
072: 変数は許されていません
073: interface が必要です
074: PC を決めてください
075: SP and Heap を決めてください
076: よけいな @ が付いています
077: PROGRAM が必要です ！
078: Pointer は許されていません(except @XXXX)
079: @xxxx はレベル１の Proc/Func 以外は使えません
080: @xxxx は引数無しの Proc/Func 以外は使えません
091: end. がファイルの最後にありません
095: Procedure のみが許されています
096: Procedure のレベルが多すぎます
097: パラメータは許されていません
098: Nest が多すぎます
100: ファイルが見つかりません
101: 無効なファイル名です
else Need more debug

７．コンパイルの例

例１：　コメントとコンパイル指令

```
{$mem $ffec04,$6000,$2000  heap,stack,pc のアドレス指定 }
program test1;
var x: array[0..$2000] of byte;

// これはコメント　一行だけのコメント
{  これもコメント　終わりは　}
(* これもコメント　終わりは *)
{$c-  これがあると　$C+ まではコメントを生成しない }
 procedure clear;
 var i: longint;
 begin
   for i:=0 to 7 do x[i]:=0;
 end; {clear}
{$c+ ここまではコメントは作られない }

 procedure clear_all;
 var i: longint;
 begin
   for i:=0 to $2000 do x[i]:=$ff;
 end; {clear_all}

begin
  clear_all;
end.
```

以下のアセンブリプログラムは上のソースをコンパイルして出力された結果です。

```
; {$mem $ffec04,$6000,$2000  heap,stack,pc のア・
; program test1;
¥ram  equ     h'00ffec04
¥stack        equ     h'00006000
¥rom  equ     h'00002000
      org     ¥rom
      dl      program,¥ram
program:mov.l #¥stack,sp
      bsr     ¥lnkfre
      bsr     ¥TEST1
      bra     ¥halt
;
      db      "H8UCMPL Ver 7.10  by @  Apr 17 2005"
```

```
¥date:        db      29
      db      "compiled: 2005/05/04 21:53:31"
;
      org     ($+15)/16*16
; var x: array[0..$2000] of byte;
X             equ     ¥stack-8206
;
; // これはコメント　一行だけのコメント
; {  これもコメント　終わりは　}
; (* これもコメント　終わりは *)
; {$c-  これがあると　$C+ まではコメントを生成し
CLEAR.9:mov.b #1,r0
      bsr     ¥slset
      subs    #4,sp
      xor.l   r3,r3
      xor.w   r0,r0
$5:   mov.b   r0,@(X,r3)
      inc.l   #1,r3
      cmp.l   #7,r3
      ble     $5
      mov.l   fp,sp
      mov.l   @sp+,fp
      rts
;
; procedure clear_all;
; var i: longint;
; begin
CLEAR_ALL.16:
      mov.b   #1,r0
      bsr     ¥slset
      subs    #4,sp
; for i:=0 to $2000 do x[i]:=$ff;
      xor.l   r3,r3
      mov.b   #255,r0
$5:   mov.b   r0,@(X,r3)
      inc.l   #1,r3
      cmp.l   #8192,r3
      ble     $5
; end; {clear_all}
      mov.l   fp,sp
      mov.l   @sp+,fp
      rts
;
; begin
¥TEST1:       mov.l   fp,@-sp
```

```
      mov.l   sp,fp
      mov.l   fp,¥ram+12
      mov.l   fp,@-sp
      sub.l   #8194,sp
; clear_all;
      bsr     CLEAR_ALL.16
; end.
      mov.l   fp,sp
      mov.l   @sp+,fp
      rts
¥eoc  equ     ($+1)/2*2
; (pas lines) 24
; (asm lines) 45-10=35
      end
```