CP(K)－O

お客様各位	CC77016 （SPX用Cコンパイラ） 制限事項 第2版	CBG－D－0003号　1/6 平成13年 6月19日 日本電気株式会社 NECエレクトロンデバイス デバイスSI事業開発本部 第二DSIグループ グループマネージャ　松川 修二

（担当：岩崎 孝雄 Tel：044-435-9427）

拝啓　貴社ますますご清栄の段、お慶び申し上げます。また、平素より弊社製品をお引き立て頂き厚く御礼申し上げます。

さて、現在出荷しておりますCC77016（DSP 用 C コンパイラ）の使用にあたっての制限事項につきまして下記にご報告申し上げます。

また、現在のところCC77016の改版修正は行いません。お客様におかれましては、本制限事項に該当する設計をされる場合、回避策をおとりいただけるようお願い申し上げます。今後とも弊社製品をご愛顧賜りますようお願い申し上げます。

敬具

記

1.「CC77016制限事項」改版履歴

	日　時	改版内容
1	H13.3.5	SBG-M-0006号 （制限内容） 1.シフト命令に関する制限 2.変数の名称に関する制限 3.ポインタの扱いに関する制限 4.割り込みルーチンに関する制限
2	H13.6.19	本版：制限事項の追加 5.関数のインライン展開機能に関する制限 6.－O0オプションの設定禁止

本文欄外の★は、本版で改訂された主な箇所を示しています。

2.制限事項（①～⑥）

①シフト命令に関する制限（注意事項）。

シフト演算を記述する場合、シフト量をマイナスや40以上の値を設定することができません。

シフト量は仕様上[0－39]以内でしかできませんが、本コンパイラはマイナスや40以上の値が設定されてもコンパイルエラーとなりません。（ワークベンチ上でアセンブル時に初めてエラーとなります）

②変数の名称に関する制限。（ワークベンチV2．4以下のバージョンで発生します）

スタティック及びローカル変数に、アセンブラの予約語(ex. call, ret)と同じ名前を付けたプログラムを、デバッグオプションを付けてコンパイルしてできたアセンブラコードがワークベンチ上でエラーとなります。

③ポインタの扱いに関する制限。

volatileポインタ変数を演算して、非volatileポインタ変数へキャストして代入する様式を記述することができません。コンパイラエラーとなります。

④割りこみルーチンに関する制限。

割り込み元のC関数内にて変更されているスタックポインタ（dp0.dp4）をそのまま割り込みルーチン内のC関数用のスタックポインタとして流用することができません。

割り込みルーチンをC言語にて開発、さらに割り込み元のルーチンもC言語開発していた場合に発生します。

補足）CC77016は以下１～３のスタックに関する仕様があります。

1. 関数引数、ローカル変数およびレジスタの退避場所としてスタックを用いる
2. スタックポインタとして、DP0(Xメモリ用), DP4(Yメモリ用)を用いる
3. スタックポインタは、C設計関数がコールされるたびにdp0,dp4の値をもとに変更される。

そのため、割り込みが発生する割り込み元スタック動作のためにdp0,dp4が変更されています。この変更されているポインタをそのまま使用すると割り込み元スタックを破壊します。

★ **⑤関数のインライン展開機能に関する制限**

CC77016コンパイル実行時に、関数のインライン展開をする機能が働いた際に、誤ったコードを生成する可能性があります。

★ **⑥-O0オプションの設定禁止**

CC77016コンパイル実行時に-O0（全ての最適化禁止オプション）をつけた場合、誤ったコードを生成する可能性があります。

3.回避手段

①シフト命令に関する制限の回避。

シフト演算使用時のシフト量を[0－39]で設定されるようプログラミング時に、ご注意ください。

②変数の名称に関する制限の回避。

「デバッグオプションを外す。」「アセンブラ予約語をスタティック及びローカル変数として使用しない。」のいずれかの方法を用いてください。

③ポインタの扱いに関する制限の回避。

該当記述を行わないで下さい。

制限例）

```
extern volatile short *p          ;  volatile  pointer  *pを宣言
   void f(void)
   {
   short *q                       ;  pointer  *qを宣言
   q = p + 1                      ;  volatile  pointerを演算（＋1）してキャスト
  }
```

回避例）

```
extern volatile short *p          ;  volatile  pointer  *pを宣言
   void f(void)
   {
   volatile short *q              ;←*qもvolatile  pointerにする
   q = p + 1                      ;
  }
```

④割りこみルーチンに関する制限の回避。

この制限を回避するには、割込みルーチン用に別のスタック領域を設け、割込みの入口で元のスタックポインタを退避/専用のスタックポインタへ切替え、割込みの出口で、スタックポインタを復帰ということをする必要があります。

（多重割り込みを許可される場合は、割り込みレベルごとに制限回避が必要です。）

具体的には

1. 領域確保
   a.割り込みルーチン用スタックエリアを定義（領域確保）
   b.スタック用ポインタ、レジスタの退避エリアを定義
2. 命令追加
   a. 割りこみルーチン内Cモジュール実行前にスタック用ポインタ、レジスタ(r0l*1,dp0,dp4)を退避処理追加
   b. 及びスタックポインタ（dp0,dp4）を割り込み用スタックエリアに変更処理追加
   c. Cモジュール実行後スタックポインタ、レジスタ（r0l*1,dp0,dp4）を復帰処理追加

を行ってください。

＊1：r0lは例示したもので汎用レジスタ（r0l～r7l）どれでも可能です。スタックポインタのメモリ退避処理で書き換わるため最初に退避させます。

□ ④制限についての回避例を（4/6～5/6）ページに記述します。

★ **⑤inline化機能に関する制限の回避。**

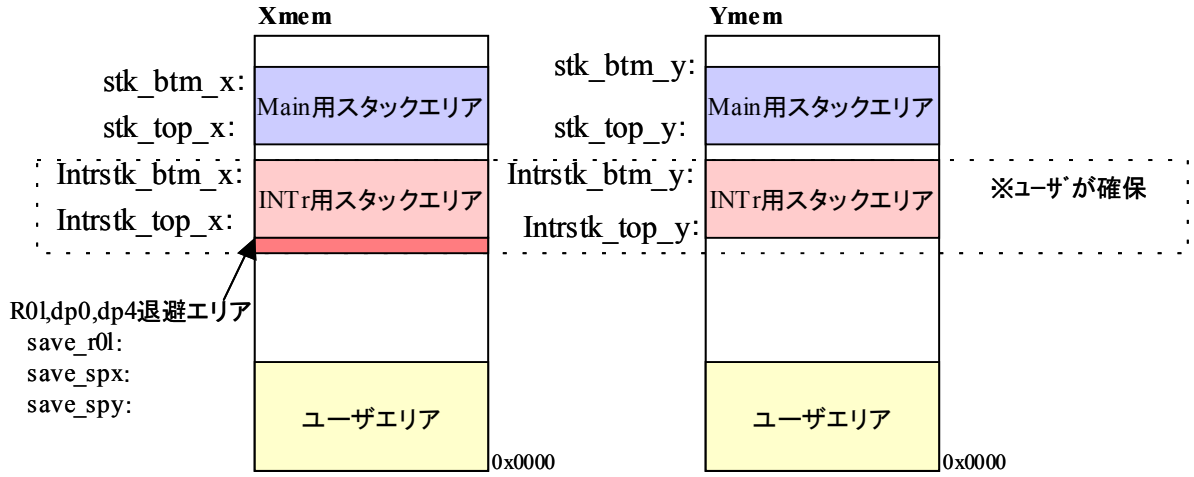
CC77016.exe実行時にその他の設定オプションに加えて、
-Osize（インライン展開を禁止するコードサイズ縮小オプション）を必ずつけ加えてください。

★ **⑥-O0オプション設定の禁止。**

-O0オプションを設定をしないで下さい。
-g（ディバグオプション），－gb，-gwgも暗黙的に-O0同様に全ての最適化を禁止いたしますが、本制限事項には該当せず、ご使用にあたっての問題はありません。

《回避例》

領域確保例

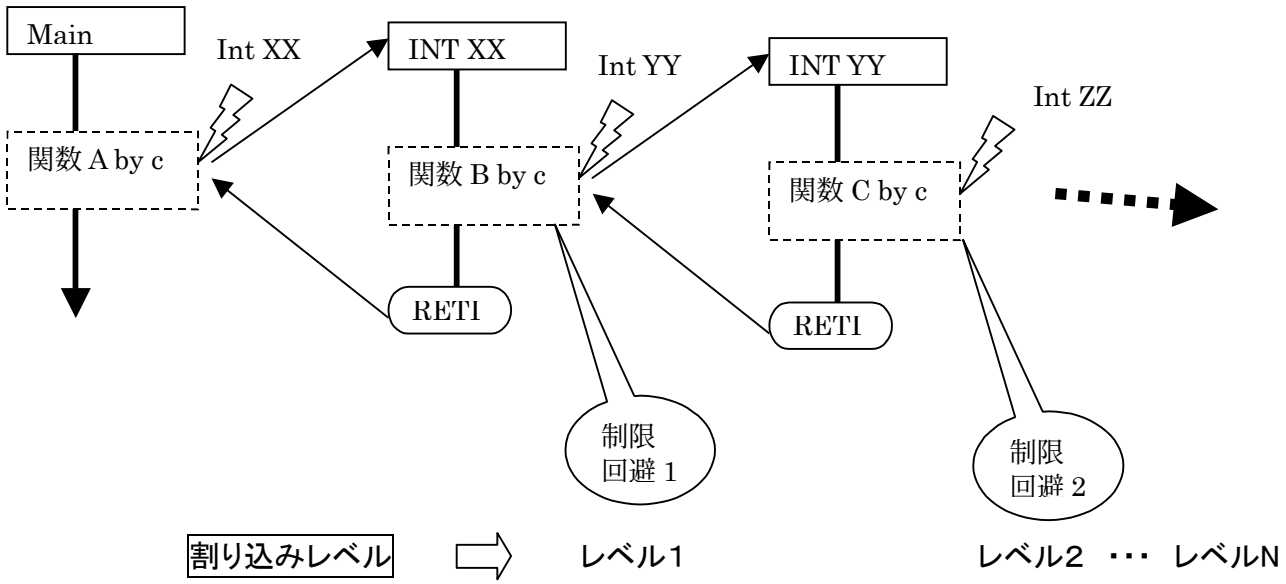



```
INTstack_X XRAMSEG
        save_r0l:       ds 1            ;r0l退避場所
        save_spx:       ds 1            ;dp0(Xメモリ用スタックポインタ)退避場所
        save_spy:       ds 1            ;dp4(Yメモリ用スタックポインタ)退避場所
        Intrstk_top_x:  ds 1024         ;割込みルーチン用スタック領域(Xメモリ)
        Intrstk_btm_x:                  ;

INTstack_Y YRAMSEG
        Intrstk_top_y:  ds 1024         ;  割込みルーチン用スタック領域(Yメモリ)
        Intrstk_btm_y:                  ;
```

割り込みプログラム例

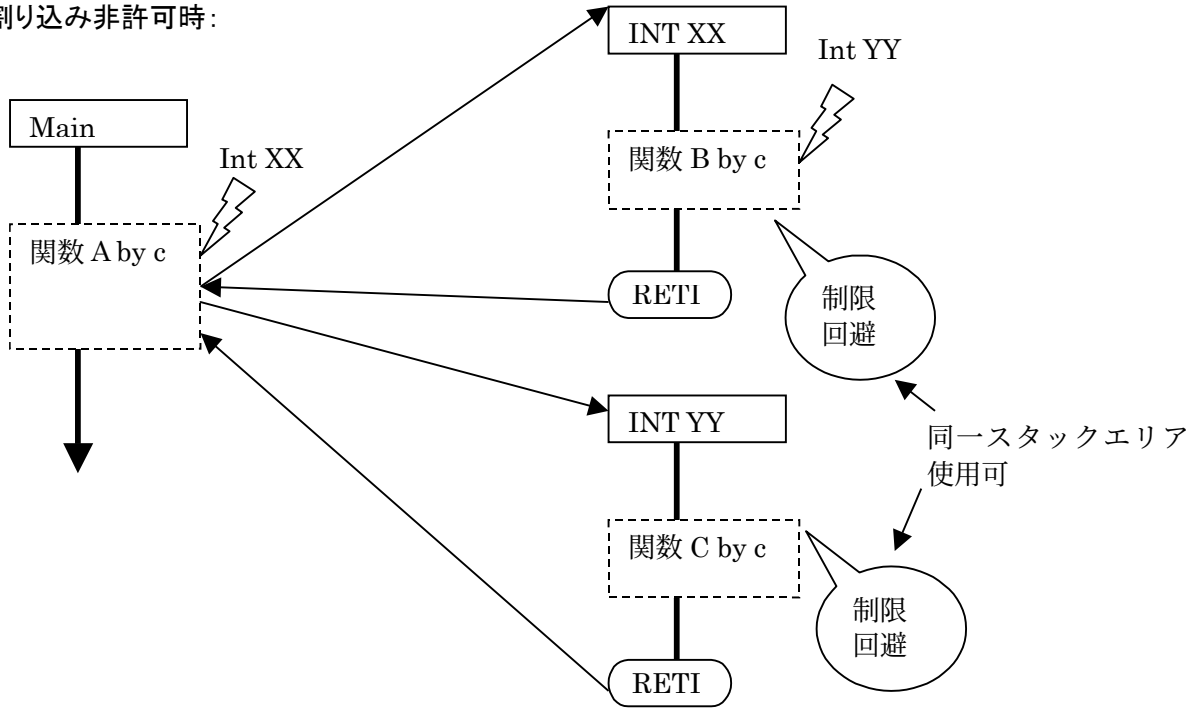
```
intr_entry:
        *save_r0l:x = r0l               ;  r0lを退避
        r0l = dp0                       ;
        *save_spx:x = r0l               ;dp0(Xメモリ用スタックポインタ)を退避
        dp0 = intr_stk_btm_x            ;割込みルーチン用スタックに切替え
        r0l = dp4                       ;
        *save_spy:x = r0l               ;dp4(Yメモリ用スタックポインタ)を退避
        dp4 = intr_stk_btm_y            ;割込みルーチン用スタックに切替え

      <必要なレジスタの退避>            ;(dp0,dp4以外のレジスタが下のcallルーチンで変更
                                        ;されたくない場合退避してください。本制限事項とは無関係)
                                        ;5/5ページご参考
        call _intr_written_by_c         ;Cで記述された割込みルーチンを実行

      <退避したレジスタの復帰>          ;(退避レジスタの復帰)

        r0l = *save_spy:x               ;
        dp4 = r0l                       ;  dp4(Yメモリ用スタックポインタ)を復帰
        r0l = *save_spx:x               ;
        dp0 = r0l                       ;  dp0(Xメモリ用スタックポインタ)を復帰
        r0l = *save_r0l:x               ;
        reti                            ;
```

補足1:多重割り込みのケース



上図のような場合、INT_XX割り込み時の関数Bのスタック処理が関数Aのスタックを破壊し、さらにINT_YY割り込み時に関数Cのスタック処理が関数Bのスタックを破壊する可能性があります。
この場合、割り込みレベル数に応じて制限事項の回避を行って下さい。
下図のように多重割り込みを許可していなければ、割り込み用スタックは共用することができます。
(メインの関数Aに対する制限のみが発生するため。)

多重割り込み非許可時:



補足2：汎用レジスタの退避について

レジスタの退避例

割り込み用に定義したスタックエリアの一部をレジスタの退避領域として利用した例を示します。

```
*dp0-- = r0e      *dp4-- = r0h      ;汎用レジスタ(r0-r7)の退避
*dp0-- = r0l      *dp4-- = r1e      ;
*dp0-- = r1h      *dp4-- = r1l      ;
.           (中略)
*dp0-- = r7h      *dp4-- = r7l      ;


r0l = dn0                           ;DNレジスタ(dn0-dn7)の退避
r1l = dn1                           ;
*dp0-- = r0l      *dp4-- = r1l      ;
           (中略)
r0l = dn6                           ;
r1l = dn7                           ;
*dp0-- = r0l          *dp4-- = r1l  ;
           (中略)
r0l = dp1                           ;DPレジスタ(dp1-dp3,dp5-dp7)の退避
r1l = dp2                           ;
*dp0-- = r0l      *dp4-- = r1l      ;
           (中略)
r0l = dp6                           ;
r1l = dp7                           ;
*dp0-- = r0l      *dp4-- = r1l      ;
```

レジスタの復帰例

最後に退避したレジスタから復帰させています。C関数スタックポインタの仕様より、最初にポインタ(dp0,dp4)をインクリメントして修正します。

```
r0l = *dp0++      r3l = *dp4++      ;ポインタ(dp0,dp4)の修正
r0l = *dp0++      r1l = *dp4++      ;最後に退避させたdp6,dp7から復帰
dp6 = r0l                           ;
dp7 = r1l                           ;
         (中略)
r1h = *dp0++      r1l = *dp4++      ;
r0l = *dp0++      r1e = *dp4++      ;
r0e = *dp0++      r0h = *dp4++      ;最初に退避させたr0,r1の復帰
```

※CC77016制限事項についてはNESDIS(NEC Semiconductor Device Information System)上に掲載しています。

-以上-