



GENTOS

Application Note #007

(AN007-V1.2)

Z
A

[GENTOS] In-line Assembly Programming using CoreRiver Compiler (`gencc`)

V1.2

August 2005

- ◆ CoreRiver Semiconductor reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice.
- ◆ Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.
- ◆ The CoreRiver Semiconductor products listed in this document are intended for usage in general electronics applications. These CoreRiver Semiconductor products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury.

Contents

1. Example of In-line Assembly Code
2. `_naked` Function
 - 1) About of "`_naked`"
 - 2) Example Function with "`_naked`"
 - 3) Example Function without "`_naked`"
3. How to Use the Label

1. Example of In-line Assembly Code

[C Code]

```
//-----  
// Sub Function  
// (C Code)  
void up_count(void) {  
    counter++;  
}
```



Modify the code.

[C Code with In-line Assembly Code]

```
//-----  
// Sub Function  
// (C Code with In-line Assembly Code)  
void up_count(void) {  
    _asm          //In-line ASM Code  
        inc counter  
        ret  
    _endasm;     //Don't forget ";"  
}
```

[In-line Assembly Programming]

- ◆ For minimum time (machine cycle), user can insert the In-line assembly code with C programming.
- ◆ `_asm` : The beginning of In-line assembly code
- ◆ `_endasm` : The end of In-line assembly code
 - ✓ Don't forget semicolon ";" after "`_endasm`"

2. `_naked` Function

- ◆ About of "`_naked`"
 - ✓ No prologue and epilogue code for the function.
 - ✓ If need, user must save any registers that may need to be preserved, selecting the proper register bank, and etc.
- ◆ This is particularly useful for interrupt functions, which can have a large (and often unnecessary) prologue and epilogue.
- ◆ Example Function with "`_naked`"

```
//-----  
// Sub Function  
// (C Code)  
void up_count(void) _naked {  
  
    counter++;  
}
```

↓ After Compiling

```
; void up_count(void) _naked  
up_count:  
    inc counter  
  
    ret
```

- ◆ Example Function without "`_naked`"

```
//-----  
// Sub Function  
// (C Code)  
void up_count(void) {  
  
    counter++;  
}
```

↓ After Compiling

```
; void up_count(void) _naked  
up_count:  
    push acc  
    push b  
    push dpl  
    push dph  
    push psw  
    mov psw, #0x00  
  
    inc counter  
  
    pop psw  
    pop dph  
    pop dpl  
    pop b  
    pop acc  
  
    ret
```

3. How to Use the Label

- ◆ In-line assembly code cannot reference the label of C code.
- ◆ C code cannot reference the label of In-line assembly code.

```
//-----  
// Sub Function  
// (C Code)  
void up_count(void) {  
    // Some C code can be here.  
  
    _asm  
        // Some ASM code can be here.  
        ljmp label_test0  
    _endasm;  
  
label_test1:    // The label of C code  
                // In-line ASM code cannot reference this label.  
  
    // Some C code can be here.  
  
    _asm  
label_test0:    // The label of In-line ASM code.  
                // C code cannot reference this label.  
  
                // Some ASM code can be here.  
    _endasm;  
  
    // Some C code can be here.  
}
```