

## HT MCU 软件堆栈的应用

文件编码：HA0001s

### 介绍：

对于 Holtek 八位元单片机来说，堆栈资源往往是有限的。例如，HT48R10A-1 就只有两级堆栈。本文将介绍如何利用软件堆栈来解决这一问题。软件堆栈是用通用数据寄存器来保存返回地址的，现在，它已经可以应用于 HT48R10A-1、HT48R30A-1、HT48R50A-1、HT48R70A-1 这几款单片机，而且，软件堆栈也将很容易应用于其他单片机。

### 用法：

为了使用软件堆栈，需要有 1 个头文件和 1 个库文件：

1. SSTACK.INC (附录一)
2. 由附录二生成的库文件 Sstack.lib。

### 步骤：

在 HT-IDE 集成环境下

1. 新建一个 project 并将 Sstack.asm 文件加到该 project 中；
2. Build 后将选择菜单 tools/library manager 在弹出的对话框中选择 open，选择路径是：x:\ide\_2000\lib\, 打入文件名：Sstack.lib 后确定，弹出的对话框问是否要新建该文件？选择确定。然后在对话框中选中新建 project 目录下的 Sstack.obj 文件点击 ADD, 然后点击 quit。(由 Sstack.asm 生成了 Sstack.lib 文件。
3. 如果要进行的项目是用以上提到的 48 系列的单片机而且需要用到软件堆栈的话, 按以下方法:
  - \*.在 OPTION/PROJECT/LIBRARY 下输入 Sstack.lib.
  - \*.在软件中定义所选择的单片机是 \_HT48R10A\_1、\_HT48R30A\_1、\_HT48R50A\_1、\_HT48R70A\_1, 例如:
 

```

                    _HT48R30A_1 EQU 1
                    
```
  - \*.通过定义 \_SS\_DEBUG EQU 1 设定调试使能
  - \*.通过 \_CALL\_DEPTH 定义堆栈级数。(默认级数为 5) 例如:
 

```

                    _CALL_DEPTH EQU 6
                    
```
  - \*.如果有超过一个的存储器指针 (MP), 则定义 \_SS\_USE\_MP1 来使用 MP1 作为堆栈指针; 否则默认使用 MP0。(对于 HT48RX0A-1 系列单片机, 只有 MP0)
  - \*.在头文件中宣告 include SSTACK.INC.
  - \*.在使用 MCALL 或 MRET 之前, 要先用 INIT\_SSTACK 指令声明。
  - \*.用 MCALL PROC\_NAME 和 MRET 来模拟 CALL 和 RET 指令。
  - \*.测试堆栈是否溢出
    - a. 定义 \_SS\_DEBUG
    - b. 将 SS\_OVERFLOW 语句写到程序中某个位置
 如果发生堆栈溢出, 用户程序会在 SS\_OVERFLOW 处停止运行。

## 例程：

本例使用 HT48R10A-1，测试三级软件堆栈。如果有超过三级的 MCALL 嵌套，程序就会停在 SS\_OVERFLOW 处。

```

;-----
;程序名: Test.asm
;作者: 盛扬半导体(上海)有限公司软件部
;目的: 测试软件堆栈
;-----
_HT48R10A_1 EQU 1 ;定义使用 HT48R10A-1 单片机
_SS_DEBUG EQU 1 ;设定调试使能
_CALL_DEPTH EQU 3 ;设定堆栈为 3 级
include SSTACK.INC
main .section at 0 'code'
    INIT_SSTACK ;初始化软件堆栈
again:
    MCALL mfunc ;调用 mfunc 子程序
    Jmp again ;循环
    SS_OVERFLOW ;如果堆栈溢出,程序停止

mfunc PROC ;mfunc2 子程序
    MCALL mfunc2 ;调用 MFUNC2 子程序
    MRET ; mfunc 子程序返回
mfunc ENDP ; mfunc 子程序结束
mfunc2 PROC
    MCALL mfunc3 ;调用 MFUNC3 子程序
    MRET ; mfunc2 子程序返回
mfunc2 ENDP ; mfunc2 子程序结束
mfunc3 PROC ; mfunc3 子程序
    MRET ; mfunc3 子程序返回
mfunc3 ENDP ; mfunc3 子程序结束
;-----
;测试程序结束
;-----

```

## 所用资源：

软件堆栈实际使用的系统资源根据所选用的单片机 ROM 页数和堆栈级数而略有区别，入下表：

资源		规格
数据寄存器		数据存储器指针 (MP) 累加器 (ACC)
数据存储器		2 × 堆栈级数 + 1
程序	INIT_STACK	2 个字节
	MRET	1 个字节

存储器	MCALL	7 个字节 (_SS_DEBUG 没有定义)
	每页	2 个字节
	其他	(6+页数) 个字节

## 使用：

本文提供两个文件，SSTACK.INC（见附录一）和 SSTACK.ASM（见附录二）。SSTACK.ASM 是用来建立库文件的（SS48R10A\_1.LIB、SS48R30A\_1.LIB、SS48R50A\_1.LIB、SS48R70A\_1.LIB）。

## 附录一：SSTACK.INC

```

;-----
;----- 软件堆栈-----
;-----
;文件名： SSTACK.INC
;介绍：
; 软件堆栈用数据存储器 (RAM) 来模拟 CALL 和 RET 指令。使用过模拟的 CALL 指
; 令后 (MCALL)，堆栈在数据存储器中从高到低排列。另外，需要使用一个数据存储器；
; 指针 (MP) 作为堆栈指针 (SP)。
; 用法：
; 1. 定义下列单片机中的一种
;    _HT48R10A_1, _HT48R30A_1, _HT48R50A_1, _HT48R70A_1
; 2. 定义 _CALL_DEPTH 来确定堆栈级数 (默认为 5 级)
;    例如：_CALL_DEPTH EQU 10H
; 3. 默认使用 MP0 作为堆栈指针
;    可以写入 _SS_USE_MP1 语句来使用 MP1 作为堆栈指针
; 4. 在使用软件堆栈之前写入 INIT_STACK 语句
; 5. 用 MCALL proc_name 和 MRET 来模拟 CALL 和 RET 指令
; 6. 检测堆栈是否溢出
;    a. 定义 SS_DEBUG
;       例如：_SS_DEBUG EQU 1
;    b. SS_OVERFLOW
;       堆栈溢出时，程序会停在 SS_OVERFLOW 处
;-----
;-----
IFNDEF    _INCLUDE_SSTACK_INC
#define    _INCLUDE_SSTACK_INC

    IFDEF    _HT48R10A_1
        _ABOVE_HT48R10A_1    EQU    1
        _SS_BOTTOM          EQU    07fh
        _SS_7bitMP          EQU    1
    ELSE

```

```

IFDEF _HT48R30A_1
  _ABOVE_HT48R10A_1 EQU 1
  _ABOVE_HT48R30A_1 EQU 1
  _SS_BOTTOM EQU 07fh
  _SS_7bitMP EQU 1
ELSE
  IFDEF _HT48R50A_1
    _ABOVE_HT48R10A_1 EQU 1
    _ABOVE_HT48R30A_1 EQU 1
    _ABOVE_HT48R50A_1 EQU 1
    _SS_BOTTOM EQU 0ffh
  ELSE
    IFDEF _HT48R70A_1
      _ABOVE_HT48R10A_1 EQU 1
      _ABOVE_HT48R30A_1 EQU 1
      _ABOVE_HT48R50A_1 EQU 1
      _ABOVE_HT48R70A_1 EQU 1
      _SS_BOTTOM EQU 0ffh
    ELSE
      MESSAGE `Software Stack: Not a valid chip'
      QUIT_SSTACK EQU 1
    ENDIF
  ENDIF
ENDIF
ENDIF
ENDIF

IFNDEF QUIT_SSTACK

IFNDEF _SS_DEBUG
  _SS_DEBUG EQU 0
ENDIF

IFNDEF _CALL_DEPTH
  _CALL_DEPTH EQU 5
ENDIF

IFNDEF _SS_USE_MP1
  _SP EQU [01h]
  _IAR EQU [00h]
ELSE
  _SP EQU [03h]
  _IAR EQU [02h]
ENDIF

```

```

_SS_TOP    EQU    _SS_BOTTOM-2*_CALL_DEPTH
_PCL      EQU    [06h]

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;INIT_STACK
;

```

```

INIT_SSTACK macro
    MOV    A,_ss_bottom-1
    MOV    _SP,A
endm

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;SS_OVERFLOW
;

```

```

SS_OVERFLOW macro
_SS_OVR_STOP:
    JMP    $
endm

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;SS_PUSH
;

```

```

SS_PUSH macro value
    MOV    A, value
    MOV    _IAR, A
    DEC    _SP
endm

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;SS_POP
;

```

```

SS_POP macro
    INC    _SP
    MOV    A, _IAR
endm

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;MCALL
;

```

```

MCALL macro label
    Local    _RETURN_ADDR
IF _SS_DEBUG
    MOV    A,_SS_TOP-1

```



```
; 3. 通过库文件管理生成 SS48RX0A_1.LIB, 并将产生的目标文件 (.OBJ) 加入
; 进去。
```

```
-----
#define SSTACK_ASM
INCLUDE SSTACK.INC

PUBLIC _SS_PAGEJMP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; INTER-PAGE JUMP

_PAGEJMPSEC .SECTION 'CODE'
_SS_PAGEJMP:
    SS_POP
    MOV    [_SS_BOTTOM], A
    SS_POP
    ADDM   A, _PCL
    JMP    _SS_PAGE0
    JMP    _SS_PAGE1
IFDEF   _ABOVE_HT48R10A_1
    JMP    _SS_PAGE2
    JMP    _SS_PAGE3
IFDEF   _ABOVE_HT48R30A_1
    JMP    _SS_PAGE4
    JMP    _SS_PAGE5
    JMP    _SS_PAGE6
    JMP    _SS_PAGE7
IFDEF   _ABOVE_HT48R50A_1
    JMP    _SS_PAGE8
    JMP    _SS_PAGE9
    JMP    _SS_PAGE10
    JMP    _SS_PAGE11
    JMP    _SS_PAGE12
    JMP    _SS_PAGE13
    JMP    _SS_PAGE14
    JMP    _SS_PAGE15
IFDEF   _ABOVE_HT48R70A_1
    JMP    _SS_PAGE16
    JMP    _SS_PAGE17
    JMP    _SS_PAGE18
    JMP    _SS_PAGE19
    JMP    _SS_PAGE20
    JMP    _SS_PAGE21
```

```

JMP    _SS_PAGE22
JMP    _SS_PAGE23
JMP    _SS_PAGE24
JMP    _SS_PAGE25
JMP    _SS_PAGE26
JMP    _SS_PAGE27
JMP    _SS_PAGE28
JMP    _SS_PAGE29
JMP    _SS_PAGE30
JMP    _SS_PAGE31

ENDIF
ENDIF
ENDIF
ENDIF

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; INTRA-PAGE JUMP
;
P0_ .SECTION AT 00FDH 'CODE'
_SS_PAGE0:
    MOV    A, [_SS_BOTTOM]
    MOV    _PCL, A
P1_ .SECTION AT 01FDH 'CODE'
_SS_PAGE1:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL, A

IFDEF _ABOVE_HT48R10A_1
P2_ .SECTION AT 02FDH 'CODE'
_SS_PAGE2:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P3_ .SECTION AT 03FDH 'CODE'
_SS_PAGE3:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A

IFDEF _ABOVE_HT48R30A_1
P4_ .SECTION AT 04FDH 'CODE'
_SS_PAGE4:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P5_ .SECTION AT 05FDH 'CODE'

```

```
_SS_PAGE5:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P6_ .SECTION AT 06FDH `CODE`
_SS_PAGE6:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P7_ .SECTION AT 07FDH `CODE`
_SS_PAGE7:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A

IFDEF _ABOVE_HT48R50A_1
P8_ .SECTION AT 08FDH `CODE`
_SS_PAGE8:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P9_ .SECTION AT 09FDH `CODE`
_SS_PAGE9:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P10 .SECTION AT 0AFDH `CODE`
_SS_PAGE10:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P11 .SECTION AT 0BFDH `CODE`
_SS_PAGE11:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P12 .SECTION AT 0CFDH `CODE`
_SS_PAGE12:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P13_ .SECTION AT 0DFDH `CODE`
_SS_PAGE13:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P14_ .SECTION AT 0EFDH `CODE`
_SS_PAGE14:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P15 .SECTION AT 0FFDH `CODE`
_SS_PAGE15:
```

```
MOV    A,[_SS_BOTTOM]
MOV    _PCL,A

IFDEF _ABOVE_HT48R70A_1
P16 .SECTION AT 10FDH `CODE'
_SS_PAGE16:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P17 .SECTION AT 11FDH `CODE'
_SS_PAGE17:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P18 .SECTION AT 12FDH `CODE'
_SS_PAGE18:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P19 .SECTION AT 13FDH `CODE'
_SS_PAGE19:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P20 .SECTION AT 14FDH `CODE'
_SS_PAGE20:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P21 .SECTION AT 15FDH `CODE'
_SS_PAGE21:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P22 .SECTION AT 16FDH `CODE'
_SS_PAGE22:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P23 .SECTION AT 17FDH `CODE'
_SS_PAGE23:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P24 .SECTION AT 18FDH `CODE'
_SS_PAGE24:
    MOV    A,[_SS_BOTTOM]
    MOV    _PCL,A
P25 .SECTION AT 19FDH `CODE'
_SS_PAGE25:
    MOV    A,[_SS_BOTTOM]
```

```

MOV    _PCL,A
P26 .SECTION AT 1AFDH `CODE`
_SS_PAGE26:
MOV    A,[_SS_BOTTOM]
MOV    _PCL,A
P27 .SECTION AT 1BFDH `CODE`
_SS_PAGE27:
MOV    A,[_SS_BOTTOM]
MOV    _PCL,A
P28 .SECTION AT 1CFDH `CODE`
_SS_PAGE28:
MOV    A,[_SS_BOTTOM]
MOV    _PCL,A
P29 .SECTION AT 1DFDH `CODE`
_SS_PAGE29:
MOV    A,[_SS_BOTTOM]
MOV    _PCL,A
P30 .SECTION AT 1EFDH `CODE`
_SS_PAGE30:
MOV    A,[_SS_BOTTOM]
MOV    _PCL,A
P31 .SECTION AT 1FFDH `CODE`
_SS_PAGE31:
MOV    A,[_SS_BOTTOM]
MOV    _PCL,A
ENDIF
ENDIF
ENDIF
ENDIF
;-----
;end of SSTACK.ASM
;-----

```

2001/8/28 : 校对人 : 陈祯

修改 :

1. 将 “ SS48C10.LIB、 SS48C30.LIB、 SS48C50.LIB、 SS48C70.LIB ” 改成 : “ 由附录二生成的库文件 Sstack.lib。 ”

2. 将 :

1. “ 将 SS48C10.LIB 、 SS48C30.LIB、 SS48C50.LIB、 SS48C70.LIB 四个库文件拷贝到 HT-IDE\LIB 子目录

下。

2. 根据所选用的单片机，在 OPTION/PROJECT/LIBRARY 下输入 SS48C10、SS48C30、SS48C50 或 SS48C70。
3. 定义所选用的单片机  
HT48C10、HT48C30、HT48C50、HT48C70
4. 通过 CALL\_DEPTH 定义堆栈级数。(默认级数为 5)
5. 如果有超过一个的存储器指针 (MP)，则定义 SS\_USE\_MP1 来使用 MP1 作为堆栈指针；否则默认使用 MP0。(对于 HT48CX0 系列单片机，只有 MP0)
6. 在头文件中宣告 include SSTACK.INC。
7. 在使用 MCALL 或 MRET 之前，要先用 INIT\_SSTACK 指令声明。
8. 用 MCALL PROC\_NAME 和 MRET 来模拟 CALL 和 RET 指令。
9. 测试堆栈是否溢出
  - a. 定义 SS\_DEBUG
  - b. 将 SS\_OVERFLOW 语句写到程序中某个位置  
如果发生堆栈溢出，用户程序会在 SS\_OVERFLOW 处停止运行。

改成：

“在 HT-IDE 集成环境下

1. 新建一个 project 并将 Sstack.asm 文件加到该 project 中；
2. Build 后将选择菜单 tools/library manager 在弹出的对话框中选择 open，选择路径是：x:\ide\_2000\lib\，打入文件名：Sstack.lib 后确定，弹出的对话框问是否要新建该文件？选择确定。然后在对话框中选中新建 project 目录下的 Sstack.obj 文件点击 ADD，然后点击 quit。(由 Sstack.asm 生成了 Sstack.lib 文件)
3. 如果要进行的项目是用以上提到的 48 系列的单片机而且需要用到软件堆栈的话，按以下方法：
  - \*在 OPTION/PROJECT/LIBRARY 下输入 Sstack.lib.
  - \*在软件中定义所选择的单片机是 HT48C10、HT48C30、HT48C50、HT48C70.例如：  
HT48C30 EQU 1
  - \*通过定义 SS\_DEBUG EQU 1 设定调试使能
  - \*通过 CALL\_DEPTH 定义堆栈级数。(默认级数为 5) 例如：  
CALL\_DEPTH EQU 6
  - \*如果有超过一个的存储器指针 (MP)，则定义 SS\_USE\_MP1 来使用 MP1 作为堆栈指针；否则默认使用 MP0。(对于 HT48CX0 系列单片机，只有 MP0)
  - \*在头文件中宣告 include SSTACK.INC。
  - \*在使用 MCALL 或 MRET 之前，要先用 INIT\_SSTACK 指令声明。
  - \*用 MCALL PROC\_NAME 和 MRET 来模拟 CALL 和 RET 指令。
  - \*测试堆栈是否溢出
    - a. 定义 SS\_DEBUG
    - b. 将 SS\_OVERFLOW 语句写到程序中某个位置  
如果发生堆栈溢出，用户程序会在 SS\_OVERFLOW 处停止运行。