

MCU Software Stack Application

D/N : HA0001E

Introduction

Holtek 8 bit controllers, like any Microcontrollers, have limited stack resources. For example, the HT48R10A-1, has only two stacks while the HT48R70A-1 has 16 stacks in total. The following notes will show how the number of effective stacks can be increased using software stacks. A software stack is a method where a common data register is used to save return addresses. Here it is applied to the HT48R10A-1, HT48R30A-1, HT48R50A-1 and HT48R70A-1 devices, however the same method can be applied to any other microcontroller.

Usage

Using software stacks, a starting file and a database file are needed :

- SSTACK.INC (appendix 1) °
- Database file Sstack.lib from appendix 2

Steps

From within the HT-IDE environment:

- Create a new project and add in the Sstack.asm
- Choose OPEN from the tools/library manager dialog box with the path x:\ide_2000\lib\ and input the file name Sstack.lib. Press YES when it pops up to ask if you are sure to create a new file and then choose the Sstack.obj file of th new project from the dialog box. Select ADD and then select QUIT. The Sstack.asm will then generate a Sstack.lib file.

- If the project uses the HT48 series microcontrollers and an increase in the number of software stacks are required, the following method can be used:
 - Input Sstack.lib at OPTION/PROJECT/LIBRARY
 - The default microcontrollers are _HT48R10A_1, _HT48R30A_1, _HT48R50A_1, _HT48R70A_1. For example:


```
_HT48R30A_1 EQU 1
```
 - Pass definition _SS_DEBUG EQU 1 set debugging enabled
 - Pass _CALL_DEPTH define stack layer (the default layer is 5) For example:


```
_CALL_DEPTH EQU 6
```
 - If it exceeds one memory pointer (MP), then define _SS_USE_MP1 to use MP1 as the stack pointer or use MP0 (for HT48RX0A-1 series microcontroller, only MP0)
 - Declare include SSTACK.INC in the starting file
 - Before using MCALL or MRET, use INIT_SSTACK instruction statement
 - Use MCALL PROC_NAME and MRET to analogize CALL and RET instruction
 - Test if the stack overflows
 - a. define _SS_DEBUG
 - b. write the SS_OVERFLOW statement into the program
 If a stack overflow occurs, the user program will stop at SS_OVERFLOW

Program

The following example uses the HT48R10A-1 to test for three levels of software stacks. If it exceeds three levels of nested MCALL, the program will stop at SS_OVERFLOW

```

;-----
;Program name: Test.asm
;Author: Holtek Semiconductor (Shanghai) Inc. Software Dept.
;Purpose: software stack test
;-----
_HT48R10A_1 EQU 1      ;definition to use HT48R10A-1 MCU
_SS_DEBUG EQU 1       ;set the debugging enabled
_CALL_DEPTH EQU 3     ;set a stack depth of 3 layers
include SSTACK.INC
main .section at 0 'code'
      INIT_SSTACK      ;initialize software stack
again:
      MCALL mfunc      ;call mfunc subprogram
      Jmp again        ;loop
      SS_OVERFLOW      ;if stack overflows, program stops

```

```

mfunc PROC                                ;mfunc2 subprogram
      MCALL mfunc2                        ;call MFUNC2 subprogram
      MRET                                ;mfunc subprogram return
mfunc ENDP                                ;mfunc subprogram over
mfunc2 PROC
      MCALL mfunc3                        ;call MFUNC3 subprogram
      MRET                                ;mfunc2 subprogram return
mfunc2 ENDP                                ;mfunc2 subprogram over
mfunc3 PROC                                ;mfunc3 subprogram
      MRET                                ;mfunc3 subprogram return
mfunc3 ENDP                                ;mfunc3 subprogram over
;-----
;program test over
;-----

```

System Resources Option

The system resources used by the software stack differs from the ROM pages and stack layers of the specific MCU. Check the table below:

System Resources Option		Spec. Detail
Data register		Memory pointer (MP), accumulator(ACC)
Data memory		2xstack level +1
ROM	INIT_STACK	2 byte
	MRET	1 byte
	MCALL	7byte(_SS_DEBUG no definition)
	Per page	2byte
	Others	(6+pages) byte

Appendix

Two reference files are provided: SSTACK.INC (appencix 1) and SSTACK.ASM (appendix 2)SSTACK.ASM is for building a database file (SS48R10A_1.LIB 、SS48R30A_1.LIB 、SS48R50A_1.LIB 、SS48R70A_1.LIB)。

Appendix 1 : SSTACK.INC

```

;-----
;software stack
;-----
;file name: SSTACK.INC
;introduction:
; software stacks use RAM to analogize CALL and RET instrucionts
; Analogized CALL
; after MCALL, stacks listed by descending order in the RAM. In addition
; another Memory pointer(MP) as stack pointer(SP) is needed

```

```

;Usage:
; 1. define one of the following MCUs
;    _HT48R10A_1, _HT48R30A_1, _HT48R50A_1, _HT48R70A_1
; 2. define_CALL_DEPTH to ensure the stack (default of 5)
;    ex._CALL_DEPTH EQU 10H
; 3. take MP0 as the stack pointer
;    SS_USE_MP1 statement permits MP1 to be used as stack pointer
; 4. input INIT_STACK statement before using the software stack
; 5. use MCALL proc_name and MRET to analogize CALL and RET
;    instruction
; 6. test if the stack overflows
;    a. define SS_DEBUG
;       ex._SS_DEBUG EQU 1
;    b. SS_OVERFLOW
;       if the stack overflows, the program stops at SS_OVERFLOW
;-----
;-----
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
ENDIF
ENDIF

```

```

IFNDEF QUIT_SSTACK

IFNDEF _SS_DEBUG
_SSTACK_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

_SSTACK_TOP EQU _SSTACK_BOTTOM-2*_CALL_DEPTH
_PCL EQU [06h]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;INIT_SSTACK
;
INIT_SSTACK macro
    MOV    A, _SSTACK_BOTTOM-1
    MOV    _SP, A
endm

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;SS_OVERFLOW
;
SS_OVERFLOW macro
_SSTACK_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
IFDEF _SS_7bitMP
    SET    [05h].7
ENDIF
    XOR    A, _SP
    SZ     [0ah].2
    JMP    _SS_OVR_STOP
ENDIF
    SS_PUSH HIGH _RETURN_ADDR
    SS_PUSH LOW  _RETURN_ADDR
    JMP    label
_RETURN_ADDR:
endm

////////////////////////////////////
;MRET
;
IFNDEF SSTACK_ASM
EXTERN _SS_PAGEJMP: NEAR
ENDIF
MRET macro
    JMP    _SS_PAGEJMP
endm

ENDIF ;
ENDIF ;
;-----
;end of SSTACK.INC
;-----

```

Appendix 2 : SSTACK.ASM

```

;-----
;file name: SSTACK.ASM
;purpose: to build the software stack database file
;steps:
; 1. choose one of the following MCUs
;   _HT48R10A_1 EQU 1
;   _HT48R30A_1 EQU 1
;   _HT48R50A_1 EQU 1
;   _HT48R70A_1 EQU 1
; 2. assemble under HT-IDE
; 3. pass database file management and generate SS48RX0A_1.LIB,
;    add the object file (.OBJ) into it
;-----
#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'
_SSPAGE0:
    MOV    A, [_SS_BOTTOM]
    MOV    _PCL, A
P1_ .SECTION AT 01FDH 'CODE'
_SSPAGE1:
    MOV    A, [_SS_BOTTOM]
    MOV    _PCL, A

IFDEF _ABOVE_HT48R10A_1
P2_ .SECTION AT 02FDH 'CODE'
_SSPAGE2:
    MOV    A, [_SS_BOTTOM]
    MOV    _PCL, A
P3_ .SECTION AT 03FDH 'CODE'
_SSPAGE3:
    MOV    A, [_SS_BOTTOM]
    MOV    _PCL, A

IFDEF _ABOVE_HT48R30A_1
P4_ .SECTION AT 04FDH 'CODE'
_SSPAGE4:
    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
;-----
```