

HT48CAx/HT48RAx 软件应用要点

文件编码：HA0076s

概述

- CALL 和 JUMP 指令会把 BP 内容载入程序计数器。
- 中断会将程序计数器的值存入堆栈。
- RETI/RET 指令会将堆栈的值载入程序计数器。

不同 bank 之间的跳转

- (a) 设定程序段在某一个 bank (定义)
ROMBANK 伪指令：ROMBANK n codesec ; n = 0~4
- (b) 将目标地址的 bank 值载入 BP
BANK 伪指令：BANK label
- (c) 跳转到目标地址
- (d) ...

范例

```

ROMBANK 0 codesec0; definition
ROMBANK 1 codesec1; definition
ROMBANK 2 codesec2; definition
codesec0 .section at 000h 'code'; BANK 0
org 0000h
    clr bp
    jmp strt
    ...
    ...
strt:
    ...
    ...
    mov a,BANK label1      ; label1 is located in BANK 1
    mov bp,a              ; load bank number of label1 to BP
    jmp label1            ; jump to label1
    ...
    ...
label0:                   ; label0 is located in BANK 0
    ...
    ...
codesec1 .section at 000h 'code'; BANK 1
    ...
    ...
    
```

```

label1:
    ...
    ...
    mov a,BANK label2    ; label2 is located in BANK 2
    mov bp,a             ; load bank number of label2 to BP
    jmp label2           ; jump to label2
    ...
    ...
codesec2 .section at 000h 'code'; BANK 2
    ...
    ...
label2:
    ...
    ...
    mov a,BANK label0    ; label0 is located in BANK 0
    mov bp,a             ; load bank number of label0 to BP
    jmp label0           ; jump to label0
    ...
    ...
end
    
```

中断服务子程序

- (a) 备份累加器
- (b) 将 BP 值传给累加器
- (c) 清除 BP
- (d) 跳转到 bank 0 的指定地址
- (e) 备份 BP
- (f) 备份其他寄存器
- (g) ...

范例

```

include ht48ca3.inc    ; use HT48CA3 MCU body
...
...
accbuf0 db ?
statusbuf0 db ?
bpbuf0 db ?
accbuf1 db ?
statusbuf1 db ?
bpbuf1 db ?
accbuf2 db ?
statusbuf2 db ?
bpbuf2 db ?
ROMBANK 0 codesec0; definition
    
```

```

ROMBANK 1 codesec1; definition
ROMBANK 2 codesec2; definition
codesec0 .section at 000h 'code'; BANK 0
org 0004h
    mov accbuf0,a
    mov a,bp
    clr bp
    jmp labelbank00          ; labelbank00 should be located in BANK 0
org 0008h
    mov accbuf1,a
    mov a,bp
    clr bp
    jmp labelbank01          ; labelbank01 should be located in BANK 0
org 000ch
    mov accbuf2,a
    mov a,bp
    clr bp
    jmp labelbank02          ; labelbank02 should be located in BANK 0
    ...
    ...
labelbank00:                ; located in BANK 0
    mov bdbuf0,a            ; backup BP
    mov a,status
    mov statusbuf0,a
    ...
    ...                    ; can call/jump to different banks labels
    ...                    ; BP should be managed
    ...
    mov a,statusbuf0        ; restore registers
    mov status,a
    mov a,bdbuf0
    mov bp,a
    mov a,accbuf0
    reti
    ...
    ...
labelbank01:                ; located in BANK 0
    mov bdbuf1,a            ; backup BP
    mov a,status
    mov statusbuf1,a
    ...
    ...                    ; can call/jump to different banks labels
    ...                    ; BP should be managed
    ...
    mov a,statusbuf1        ; restore registers
    mov status,a
    mov a,bdbuf1
    mov bp,a
    mov a,accbuf1

```

```

    reti
    ...
    ...
labelbank02:                ; located in BANK 0
    mov bpbuf2,a            ; backup BP
    mov a,status
    mov statusbuf2,a
    ...
    ...                    ; can call/jump to different banks labels
    ...                    ; BP should be managed
    ...
    mov a,statusbuf2       ; restore registers
    mov status,a
    mov a,bpbuf2
    mov bp,a
    mov a,accbuf2
    reti
    ...
    ...
codesecl .section at 000h 'code'; BANK 1
    ...
    ...
labelbank10:
    ...
    ...
codesecl2 .section at 000h 'code'; BANK 2
    ...
    ...
labelbank20:
    ...
    ...
end
    
```

不同 bank 子程序的互相调用

- (a) 设定程序段在某一个 bank (定义)
- (b) ROMBANK 伪指令 : ROMBANK n codesecl ; n = 0~4
- (c) 备份 BP
- (d) 将目标地址的 bank 值载入 BP
- (e) BANK 伪指令 : BANK label
- (f) 调用目标地址
- (g) 将当前 bank 值传给 BP
- (h) ...
- (i) 重复 : 从步骤(b) 到步骤 (j)
- (j) ...

- (k) RET
- (l) 重新装载 BP
- (m)...

范例

```

include ht48ca3.inc          ; use HT48CA3 MCU body
...
...
accbuf0 db ?
statusbuf0 db ?
bpbuf0 db ?
accbuf1 db ?
statusbuf1 db ?
bpbuf1 db ?
accbuf2 db ?
statusbuf2 db ?
bpbuf2 db ?
ROMBANK 0 codesec0; definition
ROMBANK 1 codesec1; definition
ROMBANK 2 codesec2; definition
codesec0 .section at 000h 'code'; BANK 0
org 0000h
    clr bp
    jmp strt
    ...
    ...
strt:
    ...
    ...
    mov a,bp                ; backup BP
    mov bpbuf0,a
    mov a,BANK label1      ; label1 is located in BANK 1
    mov bp,a                ; load bank number of label1 to BP
    call label1             ; jump to label1
    mov a,bpbuf0           ; restore BP
    mov bp,a
    ...
    ...
label0:                    ; label0 is located in BANK 0
    mov a,bank label0      ; new BP
    mov bp,a
    ...
    ...
    ret
    ...
    ...
codesec1 .section at 000h 'code'; BANK 1
    ...
    
```

```

...
label1:
  mov a,bank label1      ; new BP
  mov bp,a
  ...
  ...
  mov a,bp                ; BP backup
  mov bpbuf1,a
  mov a,BANK label2      ; label2 is located in BANK 2
  mov bp,a                ; load bank number of label2 to BP
  call label2             ; jump to label2
  mov a,bpbuf1
  mov bp,a
  ...
  ...
  ret
  ...
  ...
codesecc2 .section at 000h 'code'; BANK 2
...
...
label2:
  mov a,bank label2      ; new BP
  mov bp,a
  ...
  ...
  mov a,bp                ; BP backup
  mov bpbuf2,a
  mov a,BANK label0      ; label0 is located in BANK 0
  mov bp,a                ; load bank number of label0 to BP
  call label0             ; jump to label0
  mov a,bpbuf2
  mov bp,a
  ...
  ...
  ret
  ...
  ...
end

```

产生 PFD/载波信号 (不同步) 并远距离发送

- (a) PB0 设置成 PFD 输出 (掩膜选项)
- (b) 定时/计数器 0 用来产生 PFD/载波信号
- (c) 关闭定时/计数器 0
- (d) 关闭定时/计数器 0 的溢出中断
- (e) 装载 TMR0 初值, $f_{\text{PFD}} = f_{\text{SYS}} / (2^{(n+1)} * (256-m))$, $n = \text{prescaler stage (1~8)}$, $m = 0\sim 255$
- (f) 允许定时/计数器开始计数
- (g) 清除 PBC.0, 即打开 PB.0 输出功能
- (h) 置位 PB.0 ($\text{PB.0} = 1$), 即打开 PFD 输出 ($\text{PB0} = \text{载波输出}$)
- (i) 清除 PB.0 ($\text{PB.0} = 0$), 即关闭 PFD 输出 ($\text{PB0} = 0$)
- (j) ...
- (k) 定时/计数器 1 控制一帧码中各个位之间的时间长度, 以及两帧码之间的时间长度
- (l) 建议程序都放在 bank 0, 数据可以放在 bank 0, 1 和 2 (BP 的备份和重载动作可以省略)

范例 1: 脉冲串不同步 (所有程序都在 bank 0)

$f_{\text{CARRIER}} = 38\text{kHz}$, $f_{\text{SYS}} = 4\text{MHz}$, 初始值 = $256 - 4000 / (2 * 38) = 230$

定时/计数器 0 预分频阶数 $n = 1$, $f_{\text{PFD}} = 38.46\text{kHz}$

传输次序: MSB 开始

头码: 载波 = 2.4ms

bit "1": 低电平 = 600 μs , 载波 = 1200 μs

bit "0": 低电平 = 600 μs , 载波 = 600 μs

一帧码 = 50ms

→ 定时/计数器 1 溢出周期 = 600 μs

→ 50ms/600 μs = 83 单元 (每个单元 = 600 μs)

→ 2.4ms/600 μs = 4 单元 (每个单元 = 600 μs)

```
include ht48ca3.inc          ; use HT48CA3 MCU body
```

```
...
```

```
...
```

```
accbuf2 db ?
```

```
statusbuf2 db ?
```

```
bitcount db ?
```

```
bitno db ?
```

```
frame db ?
```

```

leadframe db ?
bithl db ?
byte1 db ?           ; Tx data high byte
byte0 db ?           ; Tx data low byte
txflag dbit
...
...
org 0000h
  clr bp
  jmp strt
org 0008h
  reti
org 000ch
  mov accbuf2,a
  mov a,status
  mov statusbuf2,a
  jmp label2
strt:
...
...           ; scan key
...           ; read table to datah and datah
...           ; decode transmission format
...
  clr leadframe
  clr frame
  clr bithl
  clr tmr0c
  mov a,17
  mov bitno,a           ; bit number = 16 bits + 1 (leadframe)
  mov a,230             ; fPFD = 38.46kHz = 4000/(4*(256-230)) kHz
  mov tmr0,a
  mov a,098h
  mov tmr0c,a
  clr tmr1c
  mov a,168             ; Timer/event counter 1 overflow time unit
  mov tmr1l,a           ; = 65536-(253*256+168) μs = 600 μs
  mov a,253
  mov tmr1h,a

```

```

mov a,098h
mov tmr1c,a
mov a,09h           ; disable TMR0 interrupt
mov intc,a         ; enable TMR1 interrupt
mov a,datal        ; datal is read from ROM table (TABRDC)
mov byte0,a
mov a,datah        ; datah is read from ROM table (TABRDC)
mov byte1,a
set txflag         ; enable IR data transmission
clr bitcount       ; reset IR data bit-count
...
...
label2:
mov a,bitcount
xor a,bitno
sz status.2
clr txflag
snz txflag
jmp stoptx
sz bitcount
jmp txing
txstart:
clr pbc.0          ; generate leadframe
set pb.0
inc leadframe
snz leadframe.2   ; leadframe = 2.4ms
jmp endtxbit
inc bitcount
clr leadframe     ; clear leadframe timer
jmp endtxbit
txing:             ; transmit bit data
inc bithl
mov a,bithl
xor a,01h
snz status.2
jmp chk23
clr pbc.0         ; first : low pulse = 600µs
clr pb.0

```

```

    jmp endtxbit
chk23:
    mov a,bithl
    xor a,02h
    snz status.2
    jmp chk3
    clr pbc.0                ; second : high pulse = 600µs
    set pb.0                ; third for "0" : high pulse = none
    sz byte1.7
    jmp endtxbit
    inc bitcount
chk230:
    rlc byte0
    rlc byte1
    clr bithl
    jmp endtxbit
chk3:
    clr pbc.0                ; third for "1" : high pulse = 600µs
    set pb.0
    jmp chk230
stoptx:
    clr pb.0                ; all bits are transmitted
    clr pbc.0
endtxbit:
    inc frame                ; frame timer
    mov a,frame
    sub a,83
    snz status.0
    jmp endframe
    set txflag                ; frame time-out → re-send code
    clr bitcount            ; reload data to related registers
    mov a,datal
    mov byte0,a
    mov a,datah
    mov byte1,a
    clr bithl
    clr frame                ; clear frame timer
endframe:
    
```

```

...
...
mov a,statusbuf2
mov status,a
mov a,accbuf2
reti
...
...
    
```

范例 2：脉冲串同步 (所有程序都在 bank 0)

$f_{\text{CARRIER}} = 38\text{kHz}$, $f_{\text{SYS}} = 4\text{MHz}$, 初始值 = $256 - 4000 / (2 * 38) = 230$

定时/计数器 0 预分频阶数 $n = 1$, $f_{\text{PFD}} = 38.46\text{kHz}$

传输次序：MSB 开始

头码：载波 = $2.4\text{ms} = 88$ 个脉冲 (用 INTB 计数)

bit “1”：低电平 = $600\mu\text{s}$, 载波 = 44 个脉冲 (用 INTB 计数)

bit “0”：低电平 = $600\mu\text{s}$, 载波 = 22 个脉冲 (用 INTB 计数)

一帧码 = 50ms

→ 定时/计数器 1 溢出周期 = $600\mu\text{s}$

→ $50\text{ms} / 600\mu\text{s} = 83$ 单元 (每个单元 = $600\mu\text{s}$)

→ $2.4\text{ms} / 600\mu\text{s} = 4$ 单元 (每个单元 = $600\mu\text{s}$)

```

include ht48ca3.inc          ; use HT48CA3 MCU body
...
...
accbuf1 db ?
statusbuf1 db ?
bitcount db ?
bitno db ?
frame db ?
leadframe db ?
bith1 db ?
byte1 db ?                  ; Tx data high byte
byte0 db ?                  ; Tx data low byte
intentcom db ?
txflag dbit
pulsetxing dbit
enpulse dbit
bittxing dbit              ; bit is transmitting
    
```

```

...
...
org 0000h
  clr bp
  jmp strt
org 0004h                                ; PB0/PFD connected to PF0/INTB
  sdz intentcom
  reti
  clr pb.0
  reti
org 0008h
  reti
org 000ch
  mov accbuf2,a
  mov a,status
  mov statusbuf2,a
  jmp label2
strt:
...
...                                ; scan key
...                                ; read table to datal and datah
...                                ; decode transmission format
...
clr leadframe
clr frame
clr bithl
clr intentcom
clr tmr0c
mov a,17
mov bitno,a                            ; bit number = 16 bits + 1 (leadframe)
mov a,230
mov tmr0,a
mov a,088h
mov tmr0c,a
clr tmr1c
mov a,168                              ; Timer/event counter 1 overflow time unit
mov tmr1l,a                            ; = 65536-(253*256+168) μs = 600 μs
mov a,253

```

```

mov tmr1h,a
mov a,098h
mov tmr1c,a
mov a,09h
mov intc,a           ; enable TMR1 interrupt
mov a,datal         ; datal is read from ROM table (TABRDC)
mov byte0,a
mov a,datah         ; datah is read from ROM table (TABRDC)
mov byte1,a
set txflag
clr pulsetxing
clr bitcount
clr enpulse
clr bittxing
clr pbc.0
...
...
label2:
mov a,bitcount
xor a,bitno
sz status.2
clr txflag
snz txflag
jmp stoptx
sz bitcount
jmp txing
txstart:
mov a,88           ; leadframe interrupt number
mov intentcom,a
set intc.1         ; enable external interrupt
clr intc.4         ; clear external interrupt request flag
clr intc.3         ; disable timer/event counter 1 interrupt
set intc.0         ; enable global interrupt
clr tmr0c
mov a,230         ; timer/event counter 0 synchronized
mov tmr0,a
mov a,098h
mov tmr0c,a

```

```

set pb.0                ; enable carrier output
clr pbc.0
txstart0:
sz pb.0                ; check leadframe pulses transmitted
jmp txstart1
clr intc.1             ; leadframe pulses transmitted
txstart1:
snz intc.6
jmp txstart2
clr intc.6
inc leadframe
inc frame
txstart2:
snz leadframe.2       ; leadframe = 2.4ms and 88 pulses
jmp txstart0
inc bitcount
clr leadframe         ; reset leadframe timer
jmp endtxbit
txing:                 ; transmit bit data
inc bithl
mov a,bithl
xor a,01h
snz status.2
jmp chk23
clr pbc.0             ; first : low pulse = 600µs
clr pb.0
jmp endtxbit
chk23:
mov a,bithl
xor a,02h
snz status.2
jmp chk3
mov a,22
sz byte1.7           ; bit = 0 → 22 pulses
add a,22             ; bit = 1 → 44 pulses
mov intentcom,a
set intc.1           ; enable external interrupt
clr intc.4           ; clear external interrupt request flag

```

```

clr intc.3                ; disable timer/event counter 1 interrupt
set intc.0                ; enable global interrupt
clr tmr0c
mov a,230                 ; timer/event counter 0 synchronized
mov tmr0,a
mov a,098h
mov tmr0c,a
set pb.0                  ; enable carrier output
clr pbc.0
txbit0:
sz pb.0                   ; check leadframe pulses transmitted
jmp txbit1
clr intc.1                ; leadframe pulses transmitted
txbit1:
snz intc.6
jmp txbit0
clr intc.6
inc frame
sz pb.0
jmp txbit0
txbit2:
inc bitcount
rlc byte0
rlc byte1
clr bithl
jmp label2
stoptx:
clr pb.0                  ; all bits are transmitted
clr pbc.0
endtxbit:
mov a,frame
sub a,83
snz status.0
jmp endframe
set txflag                ; frame time-out → re-send code
clr bitcount              ; reload data to related registers
mov a,datal
mov byte0,a

```

```

mov a,datah
mov byte1,a
clr bithl
clr frame           ; clear frame timer
endframe:
...
...
set intc.3
mov a,statusbuf2
mov status,a
mov a,accbuf2
reti
...
...
    
```

同步 PFD 脉冲 (产生所需要的脉冲数)

(a) 方法 1: 采用定时/计数器 0 达到同步

- 使用定时/计数器产生 PFD
- 2 个定时/计数器 0 中断溢出周期将产生一个 PFD 脉冲
- 定时/计数器 0 中断被打开
- 定时/计数器 0 从初始值开始计数以达到 PFD 同步输出, 然后打开定时/计数器 0 允许计数位
- 在定时/计数器 0 中断子程序中累计产生的脉冲, 然后关闭 PFD 输出, 即清除 PB.0

(b) 方法 2：采用外部中断达到同步

- 使用定时/计数器 0 产生 PFD
- 连接外部 PF0/INTB 管脚 和 PB0 管脚
- 1 PFD 脉冲会产生一个外部中断
- 打开定时/计数器 0
- 打开外部中断计数 PFD 脉冲数
- 在外部中断子程序中累计脉冲数，然后关闭 PFD 输出，即清除 PB.0，关闭外部中断

(c) 方法 3：采用软件指令产生脉冲

- 采用软件指令产生脉冲
- 软件指令可以控制脉冲数
- 定时/计数器可以控制载波输出周期

从程序存储器读取表格内容
(a) 设定程序段在某一个 bank (定义)

ROMBANK 伪指令：ROMBANK n codesec；n = 0~4

(b) 将目标地址的 bank 值载入 BP

BANK 伪指令：BANK label

(c) 将表格地址高字节(包括 BANK) 装入 TBHP

MOV A,HIGH table

MOV TBHP,A

(d) 将表格地址低字节装入 TBLP

MOV A,LOW table

MOV TBLP,A

(e) 使用 TABRDC 指令读取表格内容
(f) ...
范例

```
include ht48ca3.inc          ; use HT48CA3 MCU body
...
...
tablowbyte db ?             ; table low byte buffer
ROMBANK 0 codesec0; definition
ROMBANK 1 codesec1; definition
ROMBANK 2 codesec2; definition
codesec0 .section at 000h 'code'; BANK 0
org 0000h
    clr bp
```

```

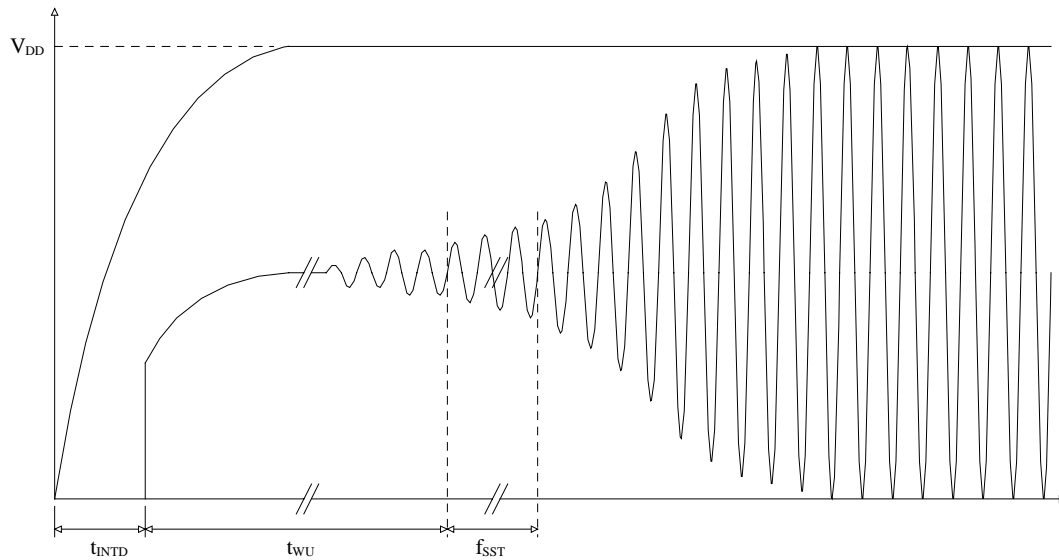
    jmp strt
    ...
    ...
strt:
    ...
    ...
    mov a,high table0      ; get bank and high byte address
    mov tbhp,a
    mov a,low table0      ; get table 0 low byte address
    mov tblp,a
    tabrdc tablowbyte     ; table read instruction
    ...
    ...
    mov a,high table1     ; get bank and high byte address
    mov tbhp,a
    mov a,low table1     ; get table 1 low byte address
    mov tblp,a
    tabrdc tablowbyte     ; table read instruction
    ...
    ...
    mov a,high table2     ; get bank and high byte address
    mov tbhp,a
    mov a,low table2     ; get table 2 low byte address
    mov tblp,a
    tabrdc tablowbyte     ; table read instruction
    ...
    ...
talbe0:                    ; table0 is located in BANK 0
    dc 0a55ah, 05aa5h    ; bank 0 table
    ...
    ...
codesecl .section at 000h 'code'; BANK 1
    ...
    ...
table1:                    ; table 1 is located in BANK 1
    dc 09669h, 06996h    ; bank 1 table
    ...
    ...
codesecl2 .section at 000h 'code'; BANK 2
    ...
    ...
table2:                    ; table 2 is located in BANK 2
    dc 02dd2h,0d22dh     ; bank 2 table
    ...
    ...
end
    
```

载波频率(f_{CARRIER}) 计算公式

$$(a) f_{\text{PFD}} = f_{\text{SYS}} / (2 * t_{\text{TMR0OV}}) = f_{\text{CARRIER}}$$

$$(b) t_{\text{TMR0OV}} = 2^{(\text{PSC2} - \text{PSC0} + 1)} * (256 - [\text{TMR0}])$$

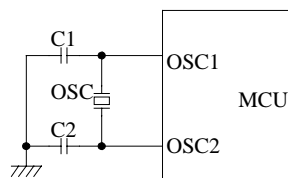
晶体振荡器结构



t_{INTD} : internal delay time and oscillator settling time

t_{wU} : crystal/resonator warm-up time(500us~5ms), dependent on crystal/resonator quality/capacitor(C1 and C2) and oscillator cease time

f_{SST} : system start-up time, $f_{\text{SC}}/1024$



f_{osc}	陶瓷谐振器		石英晶振	
	C1	C2	C1	C2
393kHz	300pF	300pF	-	-
400kHz	300pF	300pF	-	-
455kHz	300pF	300pF	-	-
480kHz	300pF	300pF	-	-
600kHz	300pF	300pF	-	-
1MHz	120pF	120pF	0pF	0pF
2MHz	0pF	0pF	0pF	0pF
3.58MHz	0pF	0pF	0pF	0pF
3.82MHz	0pF	0pF	0pF	0pF
4MHz	0pF	0pF	0pF	0pF
4.2MHz	0pF	0pF	0pF	0pF

晶体/谐振器的偏压电阻是内置的

OSC1 和 OSC2 的电容 (25pF 和 30pF) 是内置的