

## HT47R20A-1 中 A/D 转换的使用

文件编码：HA0032s

本文主要介绍 HT47R20A-1 单片机 A/D 转换器的应用及注意事项。(附带实例)

### 介绍

HT47R20A-1 有两个 RC 型的 A/D 转换通道，包含两个可编程 16 位向上计数的计数器，计数器 A 的时钟来源可以是系统时钟、指令时钟（系统时钟/4）或 RTC 中断信号，计数器 B 的时钟来源是外部 RC 振荡电路。当 ADC/TM 位为“1”时（寄存器 ADCR 的第 1 位），TMRAL、TMRAH、TMRBL、TMRBH 组成了 A/D 转换器。

A/D 转换定时器 B 的时钟来源可以来自通道 α IN0 外部时钟输入模式、RS0-CS0 振荡器、RT0-CS0 振荡器、CRT0-CS0 振荡器（CRT0 为一电阻）或 RS0-CRT0 振荡器（CRT0 为一电容）或来自通道 1（RS1-CS1 振荡器、RT1-CS1 振荡器或 IN1 外部时钟输入）。

与 A/D 转换有关的寄存器有 TMRAH、TMRAL、TMRC、TMRBH、TMRBL 和 ADCR。内部定时器时钟输入到 TMRAH 和 TMRAL 中，A/D 时钟输入到 TMRBH 和 TMRBL 中；赋初值时要先写入低字节，再写入高字节；读取时则要先读高字节，再读低字节。如果想对两组时钟计数器赋不同的初值，则一定要先将 ADCR 寄存器的第一位置为 1 后再进行操作。OVB/OVA 位（ADCR 寄存器的第 0 位）用来设置定时器 A（或定时器 B）溢出作为定时/计数器中断信号。在 A/D 转换模式下，当定时器 A（或定时器 B）溢出时 TON 位被清除并且计数器停止计数。此次 A/D 转换相应结束。

标志	位	功能
OVB/OVA	0	在 RC 型 A/D 转换模式下，该位用来定义定时/计数器中断来自定时器 A 溢出或定时器 B 溢出（0=定时器 A 溢出，1=定时器 B 溢出） 在定时/计数器模式下，该位空缺
ADC/TM	1	设定定时/计数器或 RC 型 A/D 转换器允许（0=定时/计数器允许，1=A/D 转换器允许）
—	2-3	未定义，读取时为“0”
M0 M1 M2 M3	4 5 6 7	定义 A/D 转换器的工作模式（M3, M2, M1, M0） 0000=IN0 外部时钟输入模式 0001=RS0-CS0 振荡器（参考电阻和参考电容） 0010=RT0-CS0 振荡器（传感器电阻和参考电容） 0011=CRT0-CS0 振荡器（传感器电阻和参考电容） 0100=RS0-CRT0 振荡器（参考电阻和传感器电容） 0101=RS1-CS1 振荡器（参考电阻和参考电容） 0110=RT1-CS1 振荡器（传感器电阻和参考电容） 0111=IN1 外部时钟输入模式 1xxx=未定义

ADCR 寄存器

寄存器 ADCR 的 4~7 位用来决定选取哪一组电阻、电容来组成 TMRBH 和 TMRBL 的振荡输入。  
寄存器 TMRC 的 TM0、TM1、TM2 用来决定定时器 A 的时钟来源。定时器 A 的时钟来源可以是系统

时钟、指令时钟或实时时钟超时时钟。

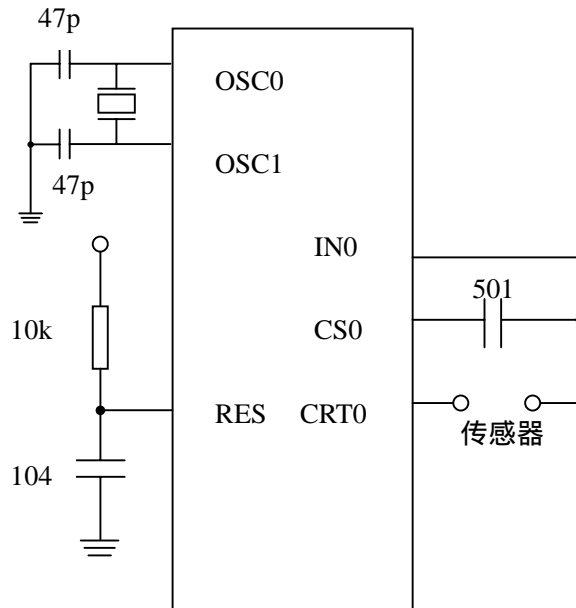
当 TON 位 (TMRC 的第 4 位) 置为 “1” 时, 定时器 A 和定时器 B 就开始计数, 直到定时器 A (或定时器 B) 发生溢出, 此时, 定时/计数器便置位中断请求标志 (TF; INTC1 的第 4 位), 同时计数器 A 和计数器 B 停止计数并 TON 位被清为 “0”。

### A/D 转换的使用

硬件部分:

传感器接至 CRT0 引脚, 电容接至 CS0 引脚

电路图如下:



软件部分:

程序中设置 TIMER A 的时钟源为系统时钟, TIMER B 对外部 RC 振荡输入计数, 并选择 TIMER A 溢出作为中断。

程序清单:

```
include ht47r20a-1.inc
data .section 'data'
count1 db ? ;count1、count2 用来保存 TIMER B 的计数值
count2 db ?

code .section at 0 'code'
org 00h
jmp start
org 04h
reti
```

```

    org    08h
    reti
    org    0ch
    reti
    org    10h
    jmp    ad_int          ; 定时/计数器中断入口
; -----
start:
    clr    intc0
    clr    intc1
    mov    a, 04h          ; 下降沿计数
    mov    tmrc, a         ; TIMER A 的时钟来源为系统时钟
    mov    a, 00110010b   ; A/D 转换允许
    mov    adcr, a        ; 并设置 TIMER B 时钟来源为 CRT0-CS0 振荡输入
    clr    acc             ; 定时/计数器赋初值
    mov    tmral, a
    mov    tmrah, a
    mov    tmrbl, a
    mov    tmrbh, a
    set    tmrc.4          ; 打开计数器, TIMER A 和 TIMER B 开始计数
    set    intc1.0        ; 定时/计数器中断允许
    set    intc0.0        ; 总中断允许
    jmp    $              ; A/D 转换进行中
; -----
; 转换结束, 进入中断程序
ad_int:                    ; 定时/计数器中断服务子程序
    mov    a, tmrbh       ; 读取 TIMER B 高字节内容
    mov    count1, a
    mov    a, tmrbl       ; 读取 TIMER B 低字节内容
    mov    count2, a
    call calculate        ; 计算
    reti
; -----
calculate proc
    ; 计算过程
    ret
calculate endp

```

### 下面介绍一种 A/D 转换的计算方法。

以第一通道为例, 当振荡源为参考电阻  $R_S$ 、参考电容  $C_S$  时, 设频率为  $f_{RSCS}$ , 有:

$$f_{RSCS} = N / (R_S \times C_S) \text{ ----- (1)}$$

$N$ ——常数

同样, 当振荡源为传感器电阻  $R_T$ 、参考电容  $C_S$  时, 有:

$$f_{RTCS} = N / (RT \times CS) \text{----- (2)}$$

设计数时间为 T，则在 T 时间内，检测到的振荡次数为

$$K = f \times T \text{----- (3)}$$

由 (1) (3) 式得：

$$K_{RSCS} = N \times T / (RS \times CS) \text{----- (4)}$$

由 (2) (3) 式得：

$$K_{RTCS} = N \times T / (RT \times CS) \text{----- (5)}$$

(4) (5) 式两边分别相比得：

$$RT = K_{RSCS} \times RS / K_{RTCS} \text{----- (6)}$$

这就是传感器电阻的计算公式，由此公式可见，传感器 RT 的测量值与参考电容无关。所以 RT 的测量值与参考电容的精度无关，与参考电阻 RS 的精度有关。由时间 T 内计到的 RS-CS 的振荡次数  $K_{RSCS}$ ，时间 T 内计到的 RT-CS 的振荡次数  $K_{RTCS}$ ，即可算出 RT 的值。

如果要测量传感器电容时，同样可得到：

$$CT = K_{RSCS} \times CS / K_{RSCS} \text{----- (7)}$$

注意，参考电容应为精密电容。

(6) 式虽然可得到 RT 的值，但是要算一个乘法、一个除法，都是比较复杂的运算，在 (6) 式中，要是能让

$$RS / K_{RTCS} = 2^n \text{----- (8)}$$

则得：

$$RT = K_{RSCS} \times 2^n \text{----- (9)}$$

下面，我们根据 HT47R20A-1 的特点，介绍一种方式，使得 RT 可以根据 (9) 式算出。

### 1、预估 RT 的值决定 n：

以 RT、CS 为振荡源，测量在时间  $T_x$  内 IN0 口输入的脉冲个数  $K_{RTCS}$ 。根据  $K_{RTCS}$  来预估 RT 的值从而决定 n 的值。

这里，n 的值主要是由用户根据所需传感器的测量速度决定的，我们建议  $n=0-8$ 。或者  $n=-1-7$ 。用户可根据自己的需要选择。这样我们可以将  $T_x$  时间内测量到的脉冲次数分成 9 档，存入表格，测量出  $K_{RTCS}$  的值时，查表就可得出对应的 n。 $T_x$  可选择在 0.5 秒之内。用户可根据需要自己定。一般只要能决定 n 的值， $T_x$  越小越好。

### 2、决定需要测量的脉冲 $K_{RTCS}$ ：

n 值定好后，由 (8) 式得：

$$K_{RTCS} = RS / 2^n \text{----- (10)}$$

四舍五入取  $RS / 2^n$  的整数为  $K_{RTCS}$ 。

现在回到 (6) 式子，RS 为已知， $K_{RTCS}$  已经找到，并且  $RS / K_{RTCS} = 2^n$ ，现在，只要找到  $K_{RSCS}$ ，就能由 (9) 式得到 RT。

这里需要注意的是，由 (3) (4) 式可以看出，测量  $K_{RTCS}$  和  $K_{RSCS}$  的时间必须一样。假设测量的时间为  $T_0$ 。

### 3、找出 $T_0$ ：

以 TIMER B 计数，TIMER A 定时，以 RT、CS 为振荡源。TIMER B 初始值为 (65536 -  $K_{RTCS}$ )，TIMER A 初始值为 0h，并设 TIMER B 溢出作为中断。发生中断后，读取 TIMER A 中的值，即为  $T_0$ 。做这些工作时，要注意定时/计数器读/写的先后顺序。

### 4、寻找 $K_{RSCS}$ ：

现在， $T_0$  已经定好了，我们就可以找出在  $T_0$  时间内，RS、CS 的振荡次数  $K_{RSCS}$ 。

以 TIMER B 计数，TIMER A 定时，以 RS、CS 为振荡源。TIMER A 初始值为 (65536 -  $T_0$ )，TIMER B

初始值为 0h，并设 TIMER A 溢出作为中断。发生中断后，读取 TIMER B 中的值，即为  $K_{RSCS}$ 。

5、得出 RT：

回到 (9) 式，所有的值都已经得到，只要将  $K_{RSCS}$  左移或右移 n 位，就可以得到 RT 的值。RT 的值得出后还没有大功告成。根据我们实验室得出的数据，即使传感器没有误差，用 HT47R20A-1 测量出来的值还是和实际的值有一定的误差。结果如下图：

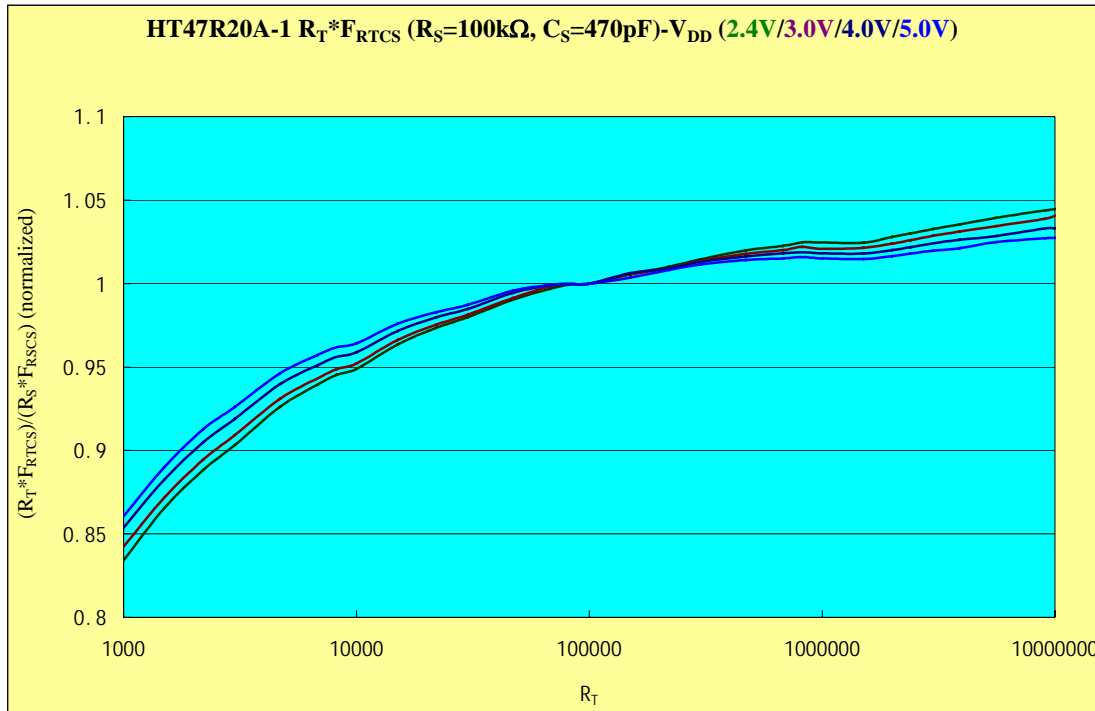


图 1：RT 测量值和实际值对照表

由表可看出，RT 在接近参考电阻 RS 的地方，误差最小，在远离参考电阻 RS 的地方误差最大。我们可以根据测量到的电阻值对照上图进行校正。

6、校正：

如图，我们将曲线分成线性的 3 段：RT 从 1000~10k、从 10k~1M、从 1M~10M。测出 RT 后，找出在三段曲线中的哪一部分，用内差法校正。这样校正要计算，会比较麻烦。另一种方法查表法可能会相对简单。

用传感器测量温度时，一般会在传感器电阻值和对应温度之间建一个表格，在建表格时，可以先考虑到 HT47R20A-1 测量误差，这样测出 RT 直接去查表就可以直接消除误差。

例程：

```
INCLUDE ht47r20a-1.inc
tempdata.section' data'
```

；计算中用到的变量

numb	db	?
data0	db	?
data1	db	?
data2	db	?
data3	db	?

```

data5      db      ?
data4      db      ?
loop       db      ?
com0       db      ?
com1       db      ?
com2       db      ?
over_flow  dbi t
temp_acc   db      ?
temp_status db      ?
; A/D 测量值存放变量
refr_h     db      ?
refr_l     db      ?
testr_h    db      ?
testr_l    db      ?

ad_bit     dbi t

over_ad    dbi t
temp_flag  dbi t

rs_h       equ 03H    ; 对应标准电阻赋初始值 03e8h(100K)
rs_l       equ 0e8h
code.sectionat 0000h' code'
; *****
;
    org     00h
    jmp     start
    org     04h
    reti
    org     08h
    reti
    org     0ch
    reti
    org     010h
; A/D 中断子程序
temper_flag_set:
    mov     temp_acc, a
    mov     a, status
    mov     temp_status, a    ; 入中断保护
    set     temp_flag        ; 一次转换结束, 可以进行下一次转换
    clr     tmrc.4
    sz     ad_bit
    jmp     ref_ad
    mov     a, tmrbh          ; RS0-CS0 振荡器组得到的值

```

```

    mov refr_h, a
    mov a, tmrbl
    mov refr_l, a
    set    ad_bit
    mov   a, temp_status
    mov   status, a
    mov   a, temp_acc
    reti

ref_ad:
    mov a, tmrbh           ; RT0~CS0 振荡器组得到的值
    mov testr_h, a
    mov a, tmrbl
    mov testr_l, a
    set    over_ad
    clr    ad_bit
    mov   a, temp_status
    mov   status, a
    mov   a, temp_acc
    reti

;-----
start:
    clr    intc0           ;
    clr    intc1

    mov   a, 3fh
    mov   mp0, a
    mov   a, 40

clr_ram:
    inc   mp0
    clr   r0
    sdz   acc
    jmp   clr_ram

    set intc0.0           ; 总中断允许
    set intc1.0           ; 计数器中断允许
    set temp_flag         ; A/D 进行中的一个标志位, temp_flag=0 表示转换进行中。初始值
                          ; 为 1

;-----
main:
    snz   temp_flag       ; temp_flag=1, 可以进行下一次转换
    jmp   main            ; ad_convert = 0 正在进行中
    sz    over_ad         ; 一次完整的操作结束 (两次 A/D 转换)

```

```

        jmp     dowi th_ad      ;去计算待测电阻值的结果

        mov     a, 12h        ;RS0~CS0 振荡器
        sz      ad_bit        ;ad_bit=0 means 正在进行参考组的计数
        mov     a, 22h        ;RT0~CS0 振荡器
        mov     adcr, a
        mov     a, 00h        ;对 timer 赋初值
        mov     tmral, a
        mov     tmrah, a
        mov     tmrbl, a
        mov     tmrbh, a
        set     tmrc.4        ;A/D 转换开始
        clr     temp_flag     ;temp_flag=0 表示转换进行中
        jmp     main         ;ad_convert = 0 正在进行中

;-----
dowi th_ad:
        clr     over_ad
        call   get_resistor_value ;一次乘法和一次除法
        jmp     $             ;得到待测的电阻值
;*****
get_resistor_value:
        mov     a, rs_h
        mov     data1, a
        mov     a, rs_l
        mov     data0, a      ;data0, data1*data2, data3=to0, to1, to2, to3
        mov     a, refr_h
        mov     data3, a
        mov     a, refr_l
        mov     data2, a

;-----
;被乘数---data1, data0 ;乘数-----data3, data2 ;乘积-----data5, data4, data3, data2
mul16:
        mov     a, 10h
        mov     loop, a
        clr     data4
        clr     data5
mul_loop:
        snz    data2.0
        jmp    no_add
        mov    a, data0
        addma, data4
        mov    a, data1
        adcma, data5

```

no\_add:

```

    clr    c
    rrc    data5
    rrc    data3
    rrc    data2
    sdz    loop
    jmp    mul_loop

```

-----

; data5, data4, data3, data2/data1, data0=data5, data4, data3, data2 remainder: com1, com2

div16:

```

    mov    a, testr_h
    mov    data1, a
    mov    a, testr_l
    mov    data0, a
    clr    com0
    clr    com1
    clr    com2
    mov    a, 32
    mov    numb, a

```

```

    clr    over_flow
    sz    data1
    jmp    no_zero
    sz    data0
    jmp    no_zero
    set    over_flow
    ret

```

no\_zero:

```

    set    c
    rlc    data2
    rlc    data3
    rlc    data4
    rlc    data5
    rlc    com2
    rlc    com1
    rlc    com0

    mov    a, com2
    sub    a, data0
    mov    com2, a
    mov    a, com1
    sbc    a, data1
    mov    com1, a

```

```
sz      c
jmp     goujian_2
sz      com0.0
jmp     goujian_2
clr     data2.0
mov     a, data0
addma, com2
mov     a, data1
adcma, com1
goujian_2:
sdz     numb
jmp     no_zero
ret
; *****
END
2003/11/18
```