

An Introduction to the A/D function in the HT47R20A-1

D/N : HA0032E

Introduction

The HT47R20A-1 contains a dual channel RC type A/D converter, inside which are two 16-bit count up timers. The clock source for Timer A can come from the system clock, the instruction clock (system clock/4), or the RTC interrupt signal. The Timer B clock comes from an external RC clock circuit. When the ADC/TM bit, which is bit 1 of the ADRC register, is set to "1", the registers TMRAL, TMRAH, TMRBL and TMRBH are used to form an A/D converter.

The A/D converter Timer/Event Counter B clock source can come from Channel 0 (IN0 external clock input mode, RS0~CS0 oscillator, RT0~CS0 oscillator, CRT0~CS0 oscillator (CRT0 as a resistor) or RS0~CRT0 oscillator (CRT0 as a capacitor), or from Channel 1, (RS1~CS1 oscillator, RT1~CS1 oscillator or IN1 external clock input)).

The registers that are connected with the A/D converter are TMRAH, TMRAL, TMRC, TMRBH, TMRBL and ADRC. The internal Timer/Event Counter clock will input to TMRAH and TMRAL, while the A/D clock will input to TMRBH and TMRBL. When writing initial values to these 16-bit timers, the low byte must first be written, after which the high byte can be written. When reading from the timers, the high byte must be read first after which the low byte can be read. If it is required to write different initial values to the two timers, then it is essential to first write a "1" to the bit 1 of the ADCR register before doing so. Bit 0 of the ADCR register, which is the OVB/ $\overline{\text{OVA}}$ bit, is used to determine which overflow, either from Timer A or Timer B, is used as the interrupt signal. When in the A/D converter mode, when either Timer A or Timer B overflows, the TON bit will be cleared and the timer

Flag	Bit	Function Description
OVB/ $\overline{\text{OVA}}$	0	When in the RC type A/D mode, this bit is used to determine if the Timer A or Timer B overflow is used as the interrupt signal. If "0", Timer A is selected, if "1" Timer B is selected.
ADC/ $\overline{\text{TM}}$	1	Either enables the Timer/Event Counter or the RC type A/D converter. If "0", the Timer/Event counter function is enabled, if "1" then the A/D converter function is enabled.
—	2~3	Undefined, will be read as "0"
M0 M1 M2 M3	4 5 6 7	Defines the A/D converter operating mode, M3, M2, M1 or M0 0000= IN0 External clock input mode 0001= RS0~CS0 oscillation – reference resistor and reference capacitor 0010= RT0~CS0 oscillation – resistor sensor and reference capacitor 0011= CRT0~CS0 oscillation – resistor sensor and reference capacitor 0100= RS0~CRT0 oscillation – reference resistor and sensor capacitor 0101= RS1~CS1 oscillation – reference resistor and reference capacitor 0110= RT1~CS1 oscillation – resistor sensor and reference capacitor 0111= IN1 external clock input mode 1xxx= Undefined

ADCR Register

The ADCR register bits 4~7 determines which resistor and capacitor group is chosen to form the TMRBH and TMRBL oscillator input.

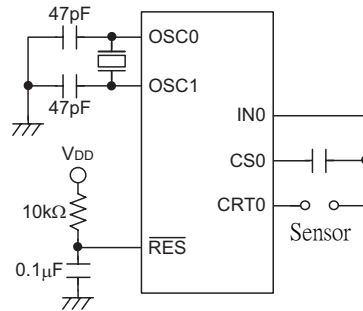
The TM0, TM1 and TM2 bits in the TMRC register are used to select the clock source for Timer/Event counter A. The clock source choices are the system clock, the instruction clock or the real time clock timeout.

When the TON bit, which is bit 4 of the TMRC register, is set to "1", Timer/Event Counter A and B will start to run. When either Timer/Event Counter A or B overflows, the interrupt request flag TF, which is bit 4 in the INTC1 register, will be set. At the same time both Timer/Event Counter A and B will stop and the TON bit will be cleared to "0".

Using the A/D Converter

- Hardware

The sensor is connected to pin CRT0 and the capacitor to pin CS0 as shown:



- Software

The clock source for Timer A is selected to be the system clock. Timer B uses an external RC clock as its clock input. Also the overflow from Timer A is chosen to be an interrupt.

- Program Example

```
include ht47r20a-1.inc
data .section 'data'
count1 db ? ;count1 and count2 stores TIMER B
count2 db ? ;count value

code .section at 0 'code'
org 00h
jmp start
org 04h
reti
org 08h
reti
org 0ch
reti
org 10h
jmp ad_int ;Timer interrupt entry point
;-----
start:
clr intc0
clr intc1
mov a,04h ;active on fallin edge
mov tmrc, a ;TIMER A clock source is the system clock
mov a,00110010b ;A/D enable
mov adcr,a ;TIMER B clock source is CRT0-CS0 osc
clr acc ;initialise counters
```

```

mov    tmral,a
mov    tmrah,a
mov    tmrbl,a
mov    tmrbh,a
set    tmrc.4           ;Timer A and Timer B start running
set    intc1.0         ;enable counter interrupt
set    intc0.0         ;global interrupt enable
jmp    $               ;A/D conversion in progress
;-----
;conversion ended, enter interrupt routine
ad_int:                ;Timer/Event counter interrupt
                        ;service routine
mov    a,tmrbh         ;read Timer B high byte
mov    count1,a
mov    a,tmrbl         ;read Timer B low byte
mov    count2,a
call   calculate       ;calculate routine
reti
;-----
calculate proc         ;calculate subroutine
ret
calculate endp

```

One Method of A/D Conversion Calculation Method

Taking the first channel as an example, when the oscillator is the reference resistor RS, the reference capacitor is CS, the frequency will be f_{RSCS}

$$f_{RSCS} = N / (RS \times CS) \text{-----(1)}$$

N - constant

Also, when the oscillator is the sensor resistor RT and the reference capacitor CS:

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

Setup an interval T, then within this time period T, the number of oscillator counts will be given by:

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

From (1) and (3) we can obtain:

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

From (2) and (3) we can obtain:

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

From both sides of (4) and (5) we can obtain:

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

This is the equation for calculating the sensor resistor. From this equation it can be seen that the value of the sensor resistance RT is not dependent upon the capacitance. Therefore the measured value of RT and the accuracy of the reference capacitor are not important. However the accuracy of the reference resistor RS is important. From the number of RS~CS oscillator counts, K_{RSCS} , within the time T, and from the number of RT~CS oscillator counts, K_{RTCS} , within the time T, the value of RT can be calculated.

If it is required to measure the sensor capacitance, in the same way we can obtain:

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

Note that the reference capacitor is a precision type.

Although we can obtain the RT value from (6), it is still necessary to conduct a multiplication and a division, both of which are not so simple. In (6) if we let:

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

We can obtain:

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

Below, using the characteristics of the HT47R20A-2 device, a method is introduced where the value as shown in (9) can be calculated.

- Estimate the RT value to determine n:
With RT and CS as the oscillator source, measure the number of pulses, K_{RT0CS} , on the input IN0 within a time T_x . According to the K_{RT0CS} estimated value of RT, the value of n can be determined.

Here, the value of n is decided by the user's sensor measurement speed requirement. It is recommended that n has a range of between 0 and 8, or 1 to 7. The user can choose according to their requirements. In this way, within the T_x time, the number of pulses divide into 9 file and save into a table. When measuring the value of K_{RT0CS} we can get the corresponding value of n from the table. T_x can chose within 0.5 seconds,. The user can define according to their needs. Generally it is only required to decide on the value of n, T_x should be as small as possible.

- Decide on the required measured pulse K_{RTCS} :
After n has been determined, from (8) we can get:

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

K_{RTCS} is taken to be the rounded up value of $RS/2^n$

Now go back to equation (6), RS is already known, K_{RTCS} has already been found, also $RS/K_{RTCS}=2^n$, Now we only need to find K_{RSCS} , and then from equation (9) we can obtain RT .

Here it is important to note that from equations (3) and (4) we can see that the measured times K_{RTCS} and K_{RSCS} should be the same. For instance if the measured time is T_0 .

- Find T_0 :
From the Timer B count and the Timer A time, and from the RT and CS as the oscillator source. The Timer B initial value is $(65536-K_{RTCS})$, and the Timer A initial value is $0H$. The overflow from Timer B is chosen to be the interrupt signal. When an interrupt happens, the value of Timer A is read which is T_0 .
- Find K_{RSCS} :
Now that T_0 has been defined, we can now find within the time period, T_0 , RS and CS oscillator counts K_{RSCS} . From the number of counts of Timer B, and the time of Timer A, and the RS and CS oscillator source. The Timer A initial value is $(65536-T_0)$, the Timer B initial value is $0H$, and the overflow from Timer A is setup as the interrupt. After an interrupt occurs, the value read from Timer B then is K_{RSCS} .

- Obtain R_T

From (9) all the values have now been obtained. It is only required to move K_{RSCS} 左 to the left or right by n bits to get the R_T value. After finally obtaining the value of R_T , the job is not over yet. According to our laboratory data, even if the sensor has no errors, the HT47R20A-1 measured values and the actual values still have a certain error. The results are shown in the table below:

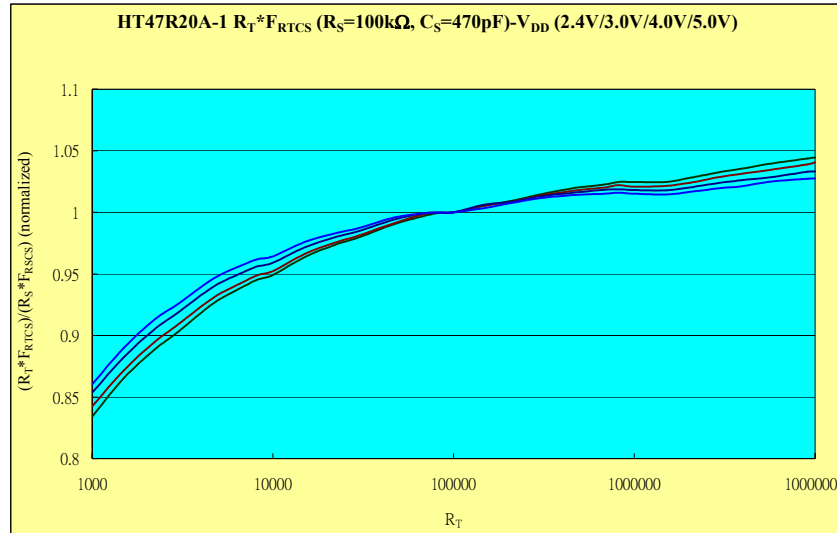


Fig. 1. R_T measured value and actual value comparison chart

From the chart, in the place where R_T is near to the reference resistor, R_S , the error is smallest. In the place where it is farthest from the reference resistor, R_S , the error is largest. Adjustment can be carried out by comparing the measured resistance value with the chart.

- Adjustment

From the chart, we divide the curve into three sections: R_T : from 1000 to 10K, from 10K to 1M and from 1M to 10M. After measuring R_T , determine which section it is in and then use interpolation to adjust. However this is quite a troublesome method. Another way is to use a table look up method, which may be simpler.

Using the sensor to measure the temperature, the measured sensor resistance and the corresponding temperature we can construct a table. When constructing this table, the HT47R20A-1 measured errors can be first considered. In this way, when the R_T value is measured, we can directly go to the table to eliminate any errors.

Example Program

```

INCLUDE ht47r20a-1.INC
tempdata .section 'data'

;Variables used in the calculations
numb          db ?
data0         db ?
data1         db ?
data2         db ?
data3         db ?
data5         db ?
data4         db ?
loop          db ?
com0          db ?
com1          db ?
com2          db ?
over_flow    dbit
temp_acc     db ?
temp_status  db ?
;A/D measurement register variables
refr_h       db ?
refr_l       db ?
testr_h      db ?
testr_l      db ?

ad_bit       dbit

over_ad      dbit
temp_flag    dbit

rs_h         equ 03H      ;corresponding reference resistor
                    ;initial value 03e8h(100K)
rs_l         equ 0e8h
code.section at 0000h 'code'
;-----
    org 00h
    jmp start
    org 04h
    reti
    org 08h
    reti
    org 0ch
    reti
    org 010h
;A/D interrupt subroutine
temper_flag_set:
    mov temp_acc, a
    mov a, status
    mov temp_status, a      ;Interrupt protection

```

```

        set temp_flag          ;single conversion finished
                                ;can now begin next conversion

        clr tmrc.4
        sz ad_bit
        jmp ref_ad
        mov a, tmrhb          ;RS0~CS0 oscillator group value
        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, tmrhb          ;RT0~CS0 oscillator obtained value
        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          ;enable master interrupt
        set intc1.0          ;enable timer interrupt
        set temp_flag        ;A/D executing flag'
                                ;temp_flag=0 indicates conversion in
                                ;progress, intital value is 1

;-----
main:
        snz temp_flag        ;temp_flag=1,execute next conversion
        jmp main            ;ad_convert=0 conversion in progress
        sz over_ad          ;first conversion operation over
                                ;two A/D conversions
        jmp dowith_ad        ;go and calculate measured res. result

```

```

mov a, 12h                ;RS0~CS0 oscillator
sz  ad_bit                ;ad_bit=0 executing reference
                               ;group count
mov a, 22h                ;RT0~CS0 oscillator
mov adcr, a
mov a, 00h                ;load initial value to timer
mov tmral, a
mov tmrah, a
mov tmrbl, a
mov tmrbh, a
set tmrc.4                ;start A/D conversion
clr temp_flag             ;temp_flag=0 conversion in progress
jmp main                   ;ad_convert=0
;-----
dowith_ad:
  clr over_ad
  call get_resistor_value ;a multiplication and a division
  jmp $                   ;to get the resistance value
;*****
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
;-----
;multiplied ---data1,data0 ;mulitplicator -----data3,data2
;Product -----data5,data4,data3,data2
mull6:
  mov a, 10h
  mov loop, a
  clr data4
  clr data5
mul_loop:
  snz data2.0
  jmp no_add
  mov a, data0
  addm a, data4
  mov a, data1
  adcm a, 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
    addm a, com2
    mov a, data1
    adcm a, com1
goujian_2:
    sdz numb
    jmp no_zero
    ret
;*****
END

```