HC08 Fan Timer

Wichit Sirichote,wichit.sirichote@gmail.com

Build a timer with Motorola Nitron MCU and using ICC08 to develop c program. Loader schematic also included. New s-record for 8-pin 68HC908QT2!

My son got his fan in the bedroom. The fan has mechanical timer for 0-180mins. One day it broken. So I got the idea to use Nitron chip to replace the mechanical timer. Someone may ask me why so complicated timer made by microcontroller chip? Actually we can build a timer with 555 and a 14-stage cmos counter! The 555 runs astable with time constant controlled by RC and for long period timing we can divide the output frequency of 555 by a cmos counter. That was my homebuilt timer 20 Yrs ago. Such circuit needs a number of passive components. We will see the hardware for timer with Motorola Nitron chip, it is only 16-pin MCU and one decoupling cap. All passive components and timing function are replaced with c coding.

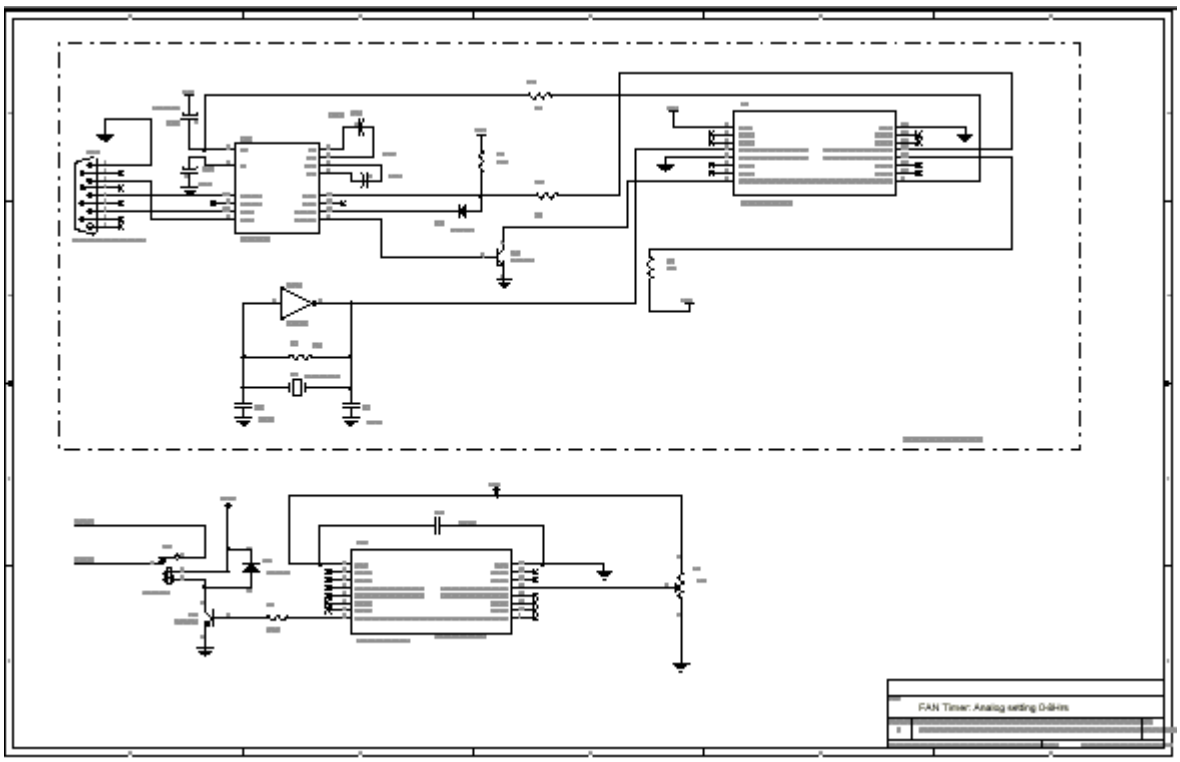

Figure 1: Complete hardware schematic: Loader and HC08 Fan Timer.

Hardware Schematic

The loader circuit is shown in the block. The hardware settings (external clock with high voltage) forces Nitron chip to run in monitor mode. The IRQ(PTA2) ties to pin 2 of MAX232 to provide HV signal. The external oscillator is required for 9600 BAUD send/receive monitor command via PTA0. It made with CMOS inverter 74HC04 and xtal 9.8304MHz. PTA1 and PTA4 also are required. The free software loader, PROG08SZ V1.7 can get from http://www.pemicro.com/.

The HC08 Fan timer circuit is shown below the loader. The Nitron chip is MC68HC908QY4, 16pin PDIP. When using the ICC08 program, you can choose the programming algorithm with QY4, say. R6 is 10k POT B type tied to the AD0, analog input channel0. It used to set time from 0Hr to 5Hrs. The output relay driving circuit is PTA3. A small NPN transistor drives a 12V electromechanical relay. D2 provides a path for back EMF current flowing when Q2 is turned off. C3 is multilayer 0.1uF decoupling cap. The Nitron chip has internal clock, power reset, low voltage detection, watchdog. You may learn more features from Niron Data sheet.

Zero Power Standby

Figure 2 shows a sample schematic featuring zero power standby. By using a push button S1 to make K1's contact closed when S1 was pressed. The K1's contact will close and latch with period setting by R6. When time-out the K1's contact will open thus turns itself power off. To restart timer, just press S1 again.

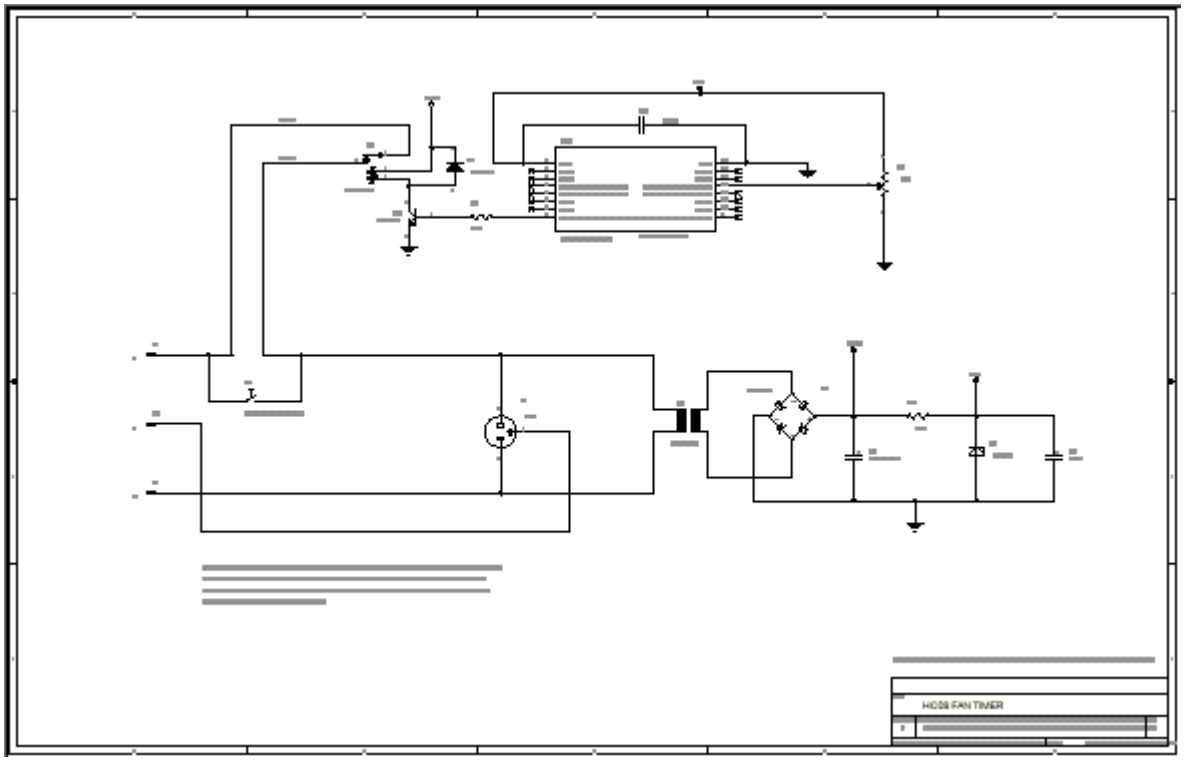

Figure 2: Sample hardware schematic with zero power standby.

Software

As shown in the schematic, we see that the timer circuit has only one 16-pin MCU, so all timing functions are done by program.

```
/*

  timer.c firmware for cheap timer using Nitron chip
  MC908QY4CP 16-pin MCU
  compiled with icc08

  Time setting is made by 10k POT, the analog input 0-5V.
  Time(mins) = (ADC reading *300)/255

  Copyright (c) 2004 Wichit Sirichote, kswichit@kmitl.ac.th

*/


#include <io908QY4.H>

#define minute 7200

// 120 ticks = 1 sec, so 7200 ticks = 1 minute

char count;
char n,timer1;
unsigned int timer2,timer3;
unsigned char sec;
unsigned int min, PV, save_time,set_time;
char x1,x2,x3;
char minute_pass;

void disable_timer()
```

```
{
   if(set_time <5) PTA &= ~0x8; // off timer
}

void minute_clock(){
   timer2++;
   if (timer2 > minute)
   {timer2 = 0;
    minute_pass = 1;
    }
}

void run_timer(){

     if (timer3 > 10)
   {
      PTA |= 0x8; // output high relay
      if (minute_pass == 1)
   {
         minute_pass = 0;
         timer3--;
      }

   }
   else
   {
        PTA &= ~0x8; // off relay
   }

}


char read_ADC()
{
   ADSCR = 0; // read channel 0 one time
  while(!(ADSCR&0x80))
  ;
  return ADR;
}
// 3-point moving average (digital filtering)

long read_ADC_filter(){
   PV = read_ADC();
   x3 = x2;
   x2 = x1;
   x1 = PV;
   PV = (x1+x2+x3)/3;
   return PV;
}

int read_time_dial(){ // from 0min to 300mins (0 to 5Hrs)
  long k;
  k = read_ADC_filter()*300/255;
  return k;
  //   return read_ADC_filter()/2; // for testing
}

// write new set time if out of set_time +-10mins
void write_time(){
  set_time = read_time_dial();
```

```
if(save_time<(set_time-10)||save_time>(set_time+10))
    {save_time = set_time;
     timer3 = save_time; // reload new set time to timer3
     timer2 = 0; // reset minute clock
 }
}

void task_led()
{
   if(++timer1>50)
  {
  timer1 = 0;
     PTA ^= 0x2; // toggle PA1
  }
}

void main()
{
  OCSTRIM = 0x81; // trim internal oscillator
  count = n = 0;
  TMODH = 0x01;
  TMODL = 0xf4;
  TSC = 0x46; // run timer
  DDRA = ~0x1; // PA0 is ADC input
  DDRB = 0xff;
  PTAPUE = ~0x81; // osc2 pin is PTA4 I/O
  ADCLK = 0x40; // ADC clock= fbus/4

 for(;;)
 {
  while(!(TSC&0x80))
  ;
  TSC &= ~0x80; // clear TOF
 // PTA ^= 2; // task running ~60Hz so the loop running is 120Hz or
1/120Hz
  task_led();
  write_time();
  minute_clock();
  disable_timer();
  run_timer();
  COPCTL = 0; // clear COP
  }
}
```

Figure 3: Firmware source code.

Main timing is done by TIM counter and TIM modulo registers, TMODL and TMODH. The preset value for both modulo registers is 0x1F4. TIM counter runs with internal bus clock / 64. Suppose the internal bus clock is 3.2MHz, so the input frequency of TIM is then 3.2MHz / 64= 50,000Hz. We need to produce system tick with 10ms or 100Hz, so the modulo register should be 500 or 0x1F4. You may see the TIM setting in the main code.

```
TMODH = 0x01;
TMODL = 0xf4;
TSC = 0x46; // run timer
```

We can test the timer overflow flag and clear it when set. The code will be,

```
while(!(TSC&0x80))
  ;
```

```
  TSC &= ~0x80; // clear TOF
```

The following tasks running below will be executed every 1/100Hz. However we can check the real timer overflow with a given output bit. I used PTA bit 1 to be toggled. I got 60Hz, so the real timing is 120Hz or 100Hz. I used 120Hz as the timing base, so one minute will be 7,200 ticks.

The ADC reading is simple code with one time conversion for channel 0. The ADC value was put into 3-point moving average filtering high frequency noise and put into the linear equation to convert ADC reading into minute value.

```
int read_time_dial(){ // from 0min to 300mins (0 to 5Hrs)
  long k;
  k = read_ADC_filter()*300/255;
  return k;
}
```

The relationship is simple, no offset, so the equation will be slope multiplied with ADC reading.

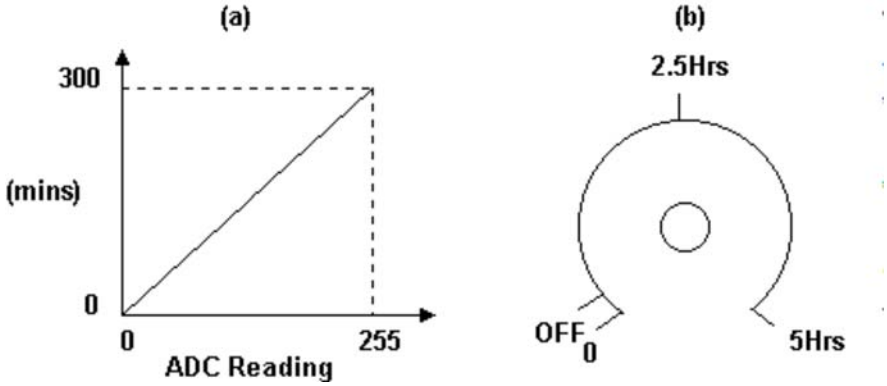

Figure 4: (a) Relationship between ADC reading and Minute (b) drawing for 10k POT.

The write time function will reload new set time if the reading was changed beyond hysteresis of +10/-10mins from the current value.

```
// write new set time if out of set_time +-10mins
void write_time(){
  set_time = read_time_dial();
if(save_time<(set_time-10)||save_time>(set_time+10))
    {save_time = set_time;
     timer3 = save_time; // reload new set time to timer3
     timer2 = 0; // reset minute clock
 }
}
```

The source code was compiled with ICC08, c compiler for HC08. You may get the full function demo version of ICC08 from imagecraft FTP, DEMO ICC08.

Schematic and s-record for 8-pin 68HC908QT2.

MC68HC908QT2-8pin MCU

I spent my weekend modify the loader circuit and recompile the source code timer.c for the 8-pin MCU, MC68HC908QT2. The MCU chip has user memory ($F800-$FDFF) only 1536 bytes. It is enough for such timer code. The source code for 16-pin timer.c was not modified. Even some code used PORTB, but it does not matter, since the 8-pin has no PORTB. With the ICC we thus need only to specify the start location of flash memory. ICC08 has the predefined for such location, so we just set the chip to QT2. It will set the start location to $F800.

For the loader, you may tie the same function of I/O pins needed for entering monitor mode. Change them from 16-pin to 8-pin, and do not forget to choose the algorithm module for QT2 in the loader software. Figure 5 shows complete hardware schematic of new timer with 8-pin 68HC908QT2. I added pilot lamp to indicate the relay has AC output and timer is running. New s-record for 8-pin MCU can be download below. Both hardware use the same source code.

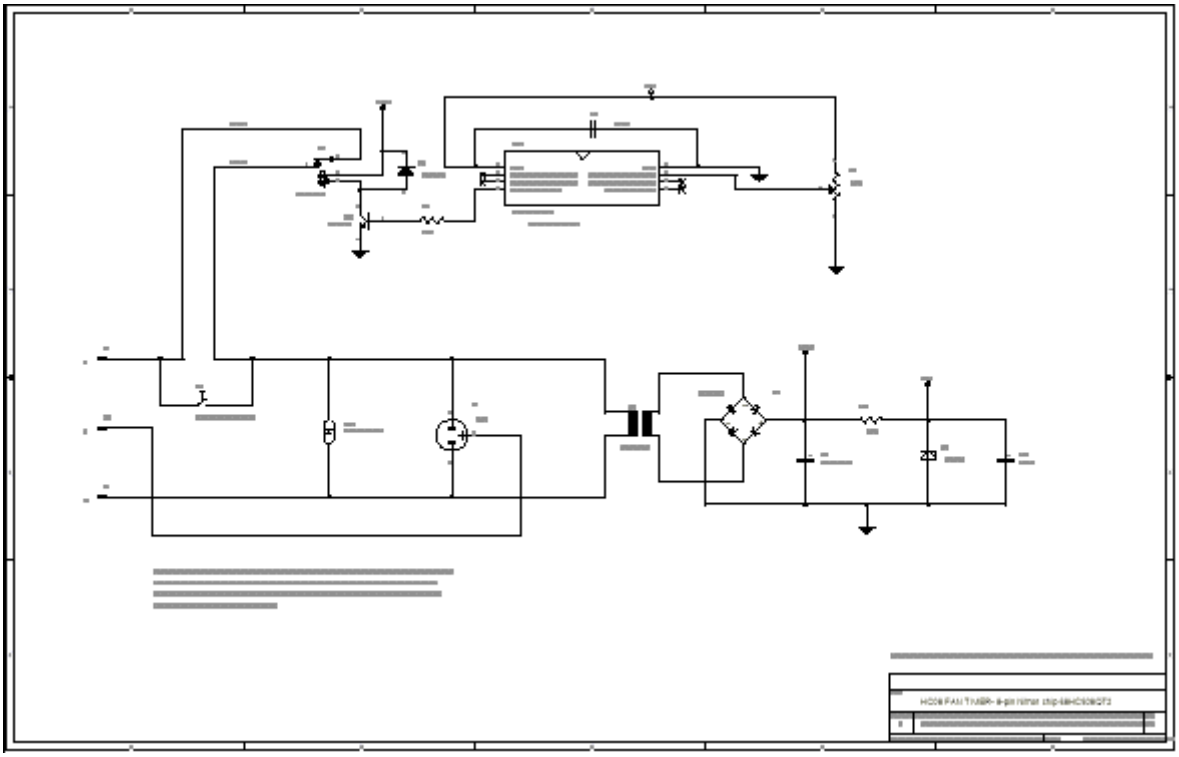

Figure 5: Hardware schematic Fan Timer using 8-pin MC68HC908QT2.

DOWNLOAD

- **Schematic: hc08fan.pdf**
- **Schematic: hc08fanqt2.pdf (for 8-pin MCU)**
- **firmware: timer.c**
- **HEX file: timer.s19**
- **HEX file: timer1.s19 (for 8-pin MCU)**

February 22, 2004