To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

H8/300L

Implementation of Real-time Clock (RTC)

Introduction

The code illustrates some of the key features to implement the real-time clock in SLP family.

The program configures Timer A as a RTC to generate an interrupt every 1 second. These interrupt increments variable seconds, minutes and hours, dates, months and years, resulting in a clock function after the initial time was set. The code also shows examples of general CPU initialization such as interrupt handling

The real-time clock (RTC) provides a binary-coded decimal (BCD) representation of the time and the date to the system's main processor. RTC accuracy is dominated by the characteristics of the external 32.768kHz "watch crystal".

The sample program is demonstrated using SLP CPU and Application board.

Target Device

H8/300L Super Low Power (SLP) series – H8/38024

Contents

1 Timer A

Timer A is an 8-bit timer with interval timing and real-time clock time-base functions.

Features of timer A are given below:

- Choice of eight internal clock sources (ø/8192, ø/4096, ø/2048, ø/512, ø/256, ø/128, ø/32, ø/8).
- Choice of four overflow periods (1 s, 0.5 s, 0.25 s, 31.25 ms) when timer A is used as a clock time base (using a 32.768 kHz crystal oscillator).
- An interrupt is requested when the counter overflows.

Use of module standby mode enables this module to be placed in standby mode independently when not used.

2 Timer Accuracy

The accuracy of timer is crystal dependent. Most other crystals tolerances are given as a +/- % of the mean or nominal value. The standard frequency tolerance is in units of parts per million (ppm), which is less cumbersome to use with 1ppm =0.0001% or 1 ppm is 1 unit referenced to 1 million units. Specifically, 1 ppm is 0.08 seconds in a 24-hour day or 1 Hz in a 1.0MHz crystal.

A crystal rated with an accuracy of 20 PPM (parts per million), can generate approximate error of about 1.7 seconds per day. For most applications, this should be adequate. It is recommended to use a crystal with higher PPM for time critical application.

2.1 How to Calculate a "ppm to Hz" Tolerance

ppm to Hz : (Tolerance in ppm) x (nominal frequency in MHz). The result is in unit if Hz.

Example: What is the minimum and maximum frequency of a 10MHz crystal with +/- 10ppm tolerance?

Tolerance = 10ppm x10MHz

= 100Hz

Max Frequency = Crystal's frequency + Tolerance

= 10 000 000Hz +100Hz

= 10 000 100Hz

Minimum Frequency = Crystal's frequency - Tolerance

=10 000 000Hz – 100Hz

= 9 999 900Hz

2.2 How to Calculate a "Hz to ppm" Tolerance

Hz to ppm : (Tolerance in Hz) / (Frequency in MHz). Result is in ppm.

Example: How many ppm is +/- 200 Hz for a 20.0 MHz crystal?

Tolerance = 200 ppm / 20MHz

= 10 ppm

Maximum ppm = Crystal's frequency + Tolerance

= 20 000 000Hz + 200Hz

= 20 000 200Hz

Minimum ppm = Crystal's frequency – Tolerance

= 20 000 000Hz – 200Hz

= 19 999 800Hz.

3 Function Overview

`void main(void)`

This main function contains the initialization of time and date upon resetting the system before starting the RTC. It will then display the counting of the RTC or date by pressing # button.

`void initialize_clock(void)`

This function will:

i) clear stand-by mode,*

ii) reset the prescaler W (PSW) and timer counter A (TCA),

iii) set the overflow period, and

iv) enable Timer A interrupt request.[@]

`void INT_TimerA(void)`

This function located in the vector table of the program (intprg.c). This function will be executed when TimerA interrupt request is generated due to overflow.

This function will:

i) clear the corresponding flag which is set to 1 in Interrupt Request Register 1 (IRR1) when timer A interrupt is requested.[#]

ii) increment and/or reset the time and date if necessary, and .

iii) check the leap year for the month of February

Notes:

*When bit 0(TACKSTP) of the clock stop register 1 (CKSTPR1) is cleared to 0, bit 3(TMA3) of the timer mode register A (TMA) cannot be written.

*Set bit 0 (TACKSTP) of the clock stop register 1(CKSTPR1) to 1 before rewriting bit 3 (TMA3) of the timer mode register A (TMA)

[@]Bit 7 of Interrupt Enable register 1 (IENR1) must be enabled to enables time A overflow interrupt requests.

[#] This flag is not cleared automatically when interrupt is accepted. Thus, it is necessary to write 0 to clear the flag.

4 Sample Code

```
/*************************************************************************
This is an example program to implement the Real-time Clock(RTC) of SLP
family.

This sample program is base on H8/38024 with SLP Series Application Board.
Minor changes might be needed on the control register setting if you are using
other microcontroller of SLP family. Please refer to the Hardware Manual for
detail.

Timer A been used for this real-clock time-base functions. The clock time-base
function is available when a 32.768kHz crystal oscillator is connected.

Interrupt Enable Register 1 (IENR1) must be enable in order to enable the
timer A overflow interrupt requests.

The input and display features are based on SLP Series Application Board. It
might be vary for different board. The functions such as
read_keypad(),SetClock() etc are only serve as a general function for
guidance. (Please refer to the zip file for full detail.)

*************************************************************************/

//Variable declaration for each of the digit in the time display
unsigned short second, minute, hour, second1, second2, minute1, minute2,
hour1,hour2;
unsigned short day, month, year, day1, day2, month1, month2, year1, year2,
year3,year4;

/* main function */
void main(void)
{
      SetClock();    // set the time for system
SetDate();     // function for initializing the time of the system using
               // 24-hour format.

initialize_clock();  // select clock time base, enable timer A interrupt etc

while(1)
{
            do
              {
              display_time();   //Time will be displayed upon calling
                                   //this function
               }while (read_keypad() != 12);   //read key pad;if button
                         //# been pressed               }while
(read_keypad() != 12);          //read key pad ;if button # been pressed

         do{
                 display_date           //Date will be shown upon
          //calling this function.
              }while (read_keypad() != 12);
}
```

```
void initialize_clock(void)
{
        P_CKSTPR1.BIT.TACKSTP = 1;       //clear standby mode

     P_TMRA.TMA.BYTE = 0x1C;            //reset PSW and TCA

        P_TMRA.TMA.BYTE = 0x18;                //select clock time base as 1s

        P_IENR1.BIT.IENTA = 1;                       //Enable Timer A
interrupt requests
}

/*****************************************************************************
This is an example program to implement the Real-time Clock(RTC) of SLP
family.

This function is located in the vector table of the program (intprg.c).

This function increment the time by 1 second with every Timer A overflow
interrupt request.

Interrupt will update the time and date accordingly.

*****************************************************************************/

__interrupt(vect=11) void INT_TimerA(void)
{
//increment 1 second with every interrupt request
   second=second+1;

  //clear the corresponding flag which is set to 1 when timer A interrupt is
  //requested.
  //this flag is not cleared automatically when interrupt is accepted.
  //Thus, it is necessary to write 0 to clear the flag.

  P_IRR1.BIT.IRRTA= 0;

  //increment minute by 1 with every 60 seconds
  //rerset minute to 0
  if(second==60)
  {
        second=0;
        minute=minute+1;
  }
  else;

  //increment hour by 1 with every 60 minutes
  //reset minute to 0
  if(minute==60)
  {
        minute=0;
        hour=hour+1;
  }
```

```
  else;

  //check the maximum day for that particular month
  //check leap year for February
  if(hour==24)
  {
       hour=0;
   }

if(month==1||month==3||month==5||month==7||month==8||month==10||month==12)
   {
       max_day=31;
   }

else if(month==2)
   {
       if(year%4==0)
        max_day=29;
    else
           max_day=28;
   }
max_day=28;   }
 else
       max_day=30;



  //set day to 1 if day=maximun day of that month
  //increment month by 1

  //increment year by 1 and set the month as 1(January) if it is 12(December)
  if(day==max_day)
  {
      day=1;

      if(month==12)
      {
           month=1;
           year=year+1;
           }
       else
             month=month+1;
  }

 else;
  }

  //if year=9999, reset year to 0
   if(year==9999)
           year=0;
   else;
}
```

5 Operating Guide

The following are the guides on how to implement the sample code given based on the H8/38024 CPU Board and the Application Board *(refer to the following figure)*:

i) Upon executing the program, LCD screen will blink.

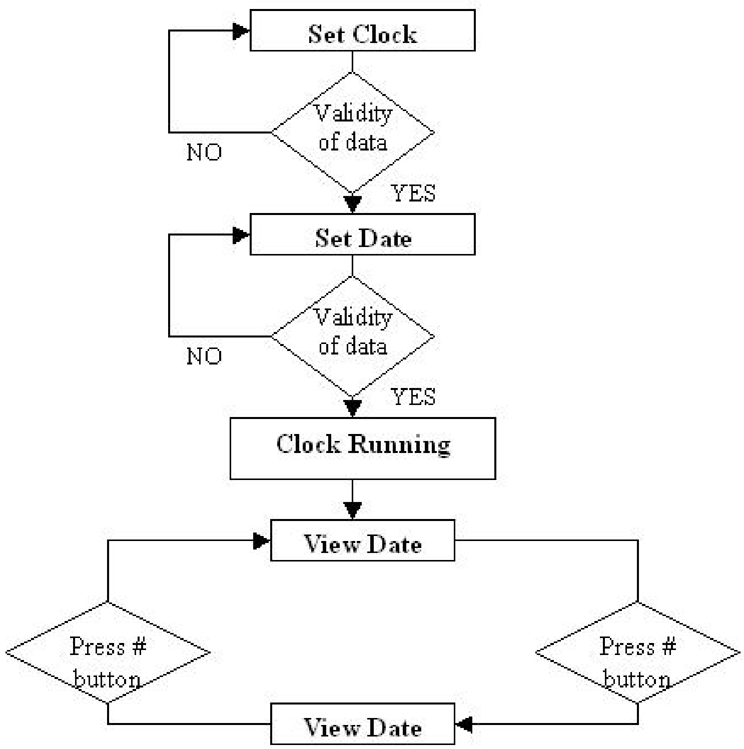
ii) Key in the time (24-hour format ~hh:mm:ss) and press "#" button to confirm.

iii) Key in the date (dd/mm/yyyy) and press "# " button to confirm

iv) Clock will start running.

v) To view the time or date, press "#" button.

The average time to execute the 1-second interrupt Timer A IRQ is 0.05 second.



Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sep.03	-	First edition issued