Application Note

AN2724
Rev. 0, 4/2004

Using the HCS12X PIT as a 24-bit Elapsed Timer

By Stephen Pickering
8/16-bit Systems Group
East Kilbride, Scotland

Introduction

This application note demonstrates how the Programmable Interrupt Timer (PIT) module on the HCS12X can be used as a 24-bit elapsed timer.

The HCS12X Programmable Interrupt Timer

The Programmable Interrupt Timer is an array of two 8-bit micro timers and four 16-bit timers that can be used to trigger peripheral modules or to raise periodic interrupts. Notable features of the Programmable Interrupt Timer are as follows.

- Four timers implemented as programmable modulus down-counters with independent timeout periods
- Timeout periods selectable between 1 and 2^{24} bus clock cycles. The timeout equals m*n bus clock cycles, where 1 <= m <= 256 and 1 <= n <= 65536
- The timers can be enabled individually
- Starting of timer channels can be aligned to each other

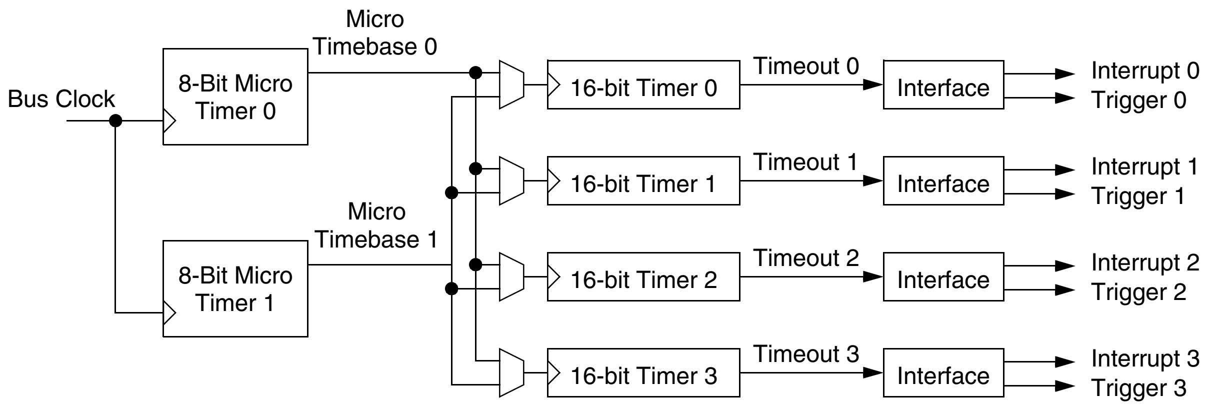

Figure 1. The HCS12X Programmable Interrupt Timer

Elapsed Time

The two micro timers can be programmed but cannot be read. Therefore, the maximum resolution is 16 bits, using the main timer.

24-bit Elapsed Time

By configuring the micro timers such that one is configured as divide-by-1 and the other as divide-by-256, and assigning the micro timers to different timers, one timer can be configured to count bus cycles modulus 256, and the other timer to count bus cycles, divided by 256 modulus, 65536. By combining the two counters, a 24-bit bus count can be achieved.

For this example, Timer 0 and Timer 1 are used with both micro timers.

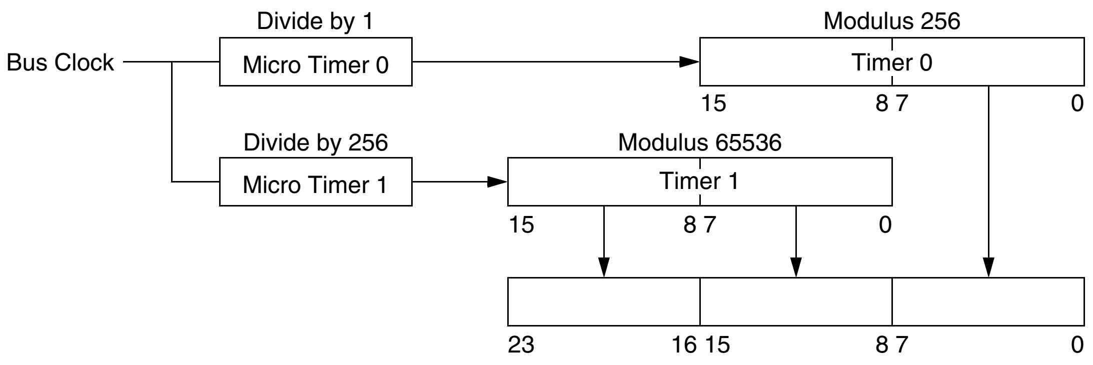

Figure 2. Logical Configuration of the Programmable Interrupt Timer

Configuring the Programmable Interrupt Timer

The code listed below will set up the Programmable Interrupt Timer as a 24-bit elapsed timer.

```
/*
Let's use the PIT as a 24 bit BUS cycle timer.
Channel 0 will be bus clocks modulo 256
Channel 1 will be bus clocks / 256
So Channel1 * 256 + Channel0 is the 24bit bus count
Micro timer0 bus clock
Micro timer1 bus clock / 256
*/
  PIT.pitcflmt.byte = PITE;       /*Enable PIT*/
  PIT.pitmtld0.byte = 0;          /*Micro timer 0 [MT0] - divide by 1*/
  PIT.pitmtld1.byte = 0xff;       /*Micro timer 1 [MT1] - divide by 256*/
  PIT.pitld0.word = 0xff;         /*Timer 0 - reload 0xff*/
  PIT.pitld1.word = 0xffff;       /*Timer 1 - reload 0xffff*/
  PIT.pitmux.byte = PMUX1;        /*Assign MT0 to Timer0 & MT1 to Timer1*/
  PIT.pitinte.byte = 0;           /*disable interrupts*/
```

Starting the timer

The code listed below can be used to start the timer

```
/*Enable timers 0 & 1*/
PIT.pitce.byte = PCE1 | PCE0;
/*load both the micro timers and the main timers as a single 16-bit*/
*((unsigned int *)&PIT.pitcflmt)=
((PITE|PITFRZ|PFLMT1|PFLMT0)<<8)|(PFLT1|PFLT0)
```

By forcing a load of both the micro timers and the main timers by a single 16-bit write to the Programmable Interrupt Timer control registers, the micro timers and main timers will be reset at the same time and will thus be in perfect synchronism.

Data

In order to use the timer, it is necessary to allocate some variables. One variable is used for a timer offset, which is an overhead associated with starting and stopping the timer. The other variable is for the number of cycles. As the cycle count can be up to 16,777,215, requiring 24 bits, 32-bit variables are used.

```
volatile unsigned long int timing_offset, cycles;
```

Stopping and Reading Time

Disabling the micro timers will stop the count, and the number of bus cycles can be read from the main timers.

```
PIT.pitcflmt.byte = PITE;/*Disable Microtimers*/
/*read timers & Calculate elapsed cycles*/
cycles = (((unsigned long)((~PIT.pitcnt1.word)&0xffff))<<8) \
       + ((unsigned long)((~PIT.pitcnt0.word)&0xff)) \
       - timing_offset;
```

It is not necessary to disable the main timers, as stopping the micro timers will prevent the main timers from advancing.

Offset

To calculate the timing offset, all that is required is to start the timer, stop it, and then read the timer's value, which is the overhead caused by starting and stopping the timer.

```
/*Enable timers 0 & 1*/
PIT.pitce.byte = PCE1 | PCE0;
/*load both the micro timers and the main timers as a single 16-bit*/
*((unsigned int *)&PIT.pitcflmt)=
((PITE|PITFRZ|PFLMT1|PFLMT0)<<8)|(PFLT1|PFLT0);
PIT.pitcflmt.byte = PITE;/*Disable Microtimers*/
/*read timers & Calculate elapsed cycles*/
timing_offset = (((unsigned long)((~PIT.pitcnt1.word)&0xffff))<<8) \
              + ((unsigned long)((~PIT.pitcnt0.word)&0xff;
```

Macros

In order to improve portability and readability the following macros can be used.

Macro	Description
TIMER_SETUP	Initializes the timer
TIMER_START	Resets the main timers (and micro timers) and synchronously starts the timers
TIMER_READ32	Stops the timer, and reads and returns the 24-bit time as a 32-bit integer
TIMER_OFFSET	Starts the timer and immediately stops it. The time recorded is the overhead associated with starting and stopping the timer
TIMER_CYCLES	Stops the timer, reads its value and subtracts the offset

```
#define TIMER_SETUP timer_setup()

#define TIMER_START  PIT.pitce.byte = PCE1 | PCE0; \
                     *((unsigned int *)&PIT.pitcflmt)= \
                     ((PITE | PITFRZ | PFLMT1 | PFLMT0)<<8 ) |(PFLT1 | PFLT0)

#define TIMER_READ32(cycles) PIT.pitcflmt.byte = PITE;/*Disable Microtimers*/ \
                             /*read timers*/ \
                             cycles = \
                            (((unsigned long)((~PIT.pitcnt1.word)&0xffff))<<8)\
                             + ((unsigned long)((~PIT.pitcnt0.word)&0xff))

#define TIMER_OFFSET(timing_offset) TIMER_START; \
                                          TIMER_READ32(timing_offset)

#define TIMER_CYCLES(cycles)  TIMER_READ32(cycles); cycles-=timing_offset
```

The timer initialization code is listed below.

```
void timer_setup(void){
/*
Let's use the PIT as a 24 bit BUS cycle timer.
Channel 0 will be bus clocks modulo 256
Channel 1 will be bus clocks / 256
So Channel1 * 256 + Channel0 is 24bit bus count
Micro timer0 bus clock
Micro timer1 bus clock / 256
*/
  PIT.pitcflmt.byte = PITE;      /*Enable PIT*/
  PIT.pitmtld0.byte = 0;         /*Micro timer 0 - divide by 1*/
  PIT.pitmtld1.byte = 0xff;      /*Micro timer 1 - divide by 256*/
  PIT.pitld0.word = 0xff;        /*Timer 0 - reload 0xff*/
  PIT.pitld1.word = 0xffff;      /*Timer 1 - reload 0xffff*/
  PIT.pitmux.byte = PMUX1;       /*Assign MT0 to Timer0 & MT1 to Timer1*/
  PIT.pitinte.byte = 0;          /*disable interrupts*/
  PIT.pitce.byte = PCE1 | PCE0;  /*Enable timer 0 & 1*/
}
```

Using the Macros

To use the macros to time code would require code similar to that shown below.

```
volatile unsigned long int timing_offset, cycles;

void main(){
  TIMER_SETUP;
  TIMER_OFFSET(timing_offset);

  TIMER_START;
  /* code */
  TIMER_CYCLES(cycles); /*read the number of elapsed cycles*/

}
```

NOTE: *It may be necessary to modify the code if timers 2 and 3 are also being used.*

References

1. HCS12X Programmable Interrupt Timer Block Guide (S12PITV1)

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com