ANALOG DEVICES

AN-765
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106 • Tel: 781/329-4700 • Fax: 781/326-8703 • www.analog.com

Measuring Duration of a Short Pulse on the ADuC702x Family

by Aude Richard

INTRODUCTION

The ADuC702x has four external interrupts which are configurable only as level triggered and active low. Therefore measuring a short pulse using the external interrupts would require some combination of external glue logic.

This application note describes a method to allow the measurement of a short pulse duration (a few milliseconds) using timer1 and the PLA. This technique does not require any external digital logic.

CONCEPT

The PLA (programmable logic array) can be described as glue logic; its function is to remove the requirement for simple external logic. It consists of 16 elements, each element containing a two-input lockup table that can be configured to generate any logic function based on one or two inputs.

The PLA can be routed to the internal interrupt system and has two dedicated interrupt bits in the interrupt controller.

Timer1 is a general-purpose timer with extra features, such as a capture event mode. Timer1 capture register (T1CAP) can be triggered by a selected interrupt source. This feature can be used to determine the start of an event with more accuracy than if a timer was triggered upon entry to the ISR. The capture event feature is used in this application note.

HARDWARE CONSIDERATION

Any of the GPIO available as PLA input can be used to measure the pulse. This method uses two elements to utilize two interrupt sources, as shown in Figure 1.

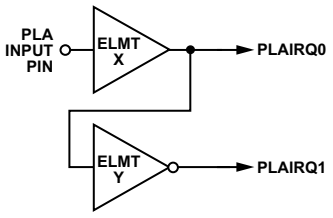

Figure 1. PLA Configuration

The pulse is passed through element x. The output of element x is configured to trigger PLAIRQ0 when its input goes high.

Output of element x is fed back to element y, configured as a "not" gate. The output of element y is configured to trigger PLAIRQ1 when the input signal goes back low.

SOFTWARE

Everything is done in the interrupt service routine, as shown in Figure 2.

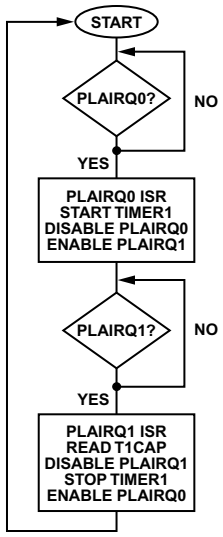

Figure 2. Flowchart

Timer1 is started in PLAIRQ0 ISR. PLAIRQ0 is disabled to ensure the ISR cannot be re-entered and PLAIRQ1 is enabled.

In the PLAIRQ1 ISR, the value captured automatically by timer1 is read in T1CAP. PLAIRQ1 is disabled to ensure the ISR cannot be re-entered and the PLAIRQ0 is re-enabled to allow a new measurement. Timer1 also needs to be stopped and reset.

See the source code provided.

LIMITATION AND PRECISION

PLAIRQ0 interrupt introduces a delay to start timer1. Interrupt latency on the ADuC702x is between 5 and 50 processor cycles, which corresponds to just over 1.1 μs in the example (using a continuous 45 MHz processor clock). Therefore, this method should only be used for pulses in the millisecond range and greater.

```
#include<aduc7020.h>

long pulse;

void My_IRQ_Function(void);       // IRQ Function Prototype

int main (void) {

  IRQ = My_IRQ_Function;          // Specify Interrupt Service Routine

  PLAELM0 = 0x0035;               // pass
  PLAELM1 = 0x0047;               // not
  PLAIRQ = 0x1110;                //

  IRQEN = 0x080000;               // enable PLA IRQ0

  while (1){
  }
}

/*********************************************************************/
/*                                                                   */
/*     Interrupt Service Routine                                     */
/*                                                                   */
/*********************************************************************/

void My_IRQ_Function()
{
  if ((IRQSTA & PLA_IRQ0_BIT) == 0x00080000)    // PLAIRQ0
  {
    T1CON = 0x32180;              // start Timer1. capture PLAIRQ1
    IRQCLR = 0x80000;             // disable PLA IRQ0
    IRQEN = 0x00100000;           // enable PLA IRQ1
  }
  if ((IRQSTA & PLA_IRQ1_BIT) == 0x00100000)    // PLAIRQ1
  {
    pulse = T1CAP;                // read the capture event
    IRQCLR = 0x00100000;          // disable PLA IRQ1
    IRQEN = 0x80000;              // enable PLA IRQ0
    T1LD = 0x00;                  // to reset timer1
    T1CON = 0xC0;                 // reset timer1
    T1CON = 0;                    // stop timer1
  }
  return ;
}
```

AN05319–0–2/05(0)