```
/*************************************************************************
 The program inputs data through port A
          in nonlatched mode and then outputs the data to port B single
          buffered every 5 seconds.

How is it done?

1. Initialize the PI/T for nonlatched input (mode 0) Port A; and
   single buffered output thru Port B.

2. To maintain 5 secs interval, the timer is programmed in to count 5
   secs in the interrupt mode.

3. To coincide the I/O with the 5 secs mark, the I/O is done within the
   interrupt service routine.
 *************************************************************************/

/* Timer Register Addresses */

#define tmr  ((unsigned char*) 0xFE8021)     /* Timer Base Address         */
#define tcr  (( unsigned char*) tmr)         /* Timer Control Reg          */
#define tivr (( unsigned char*) tmr+2)       /* Timer Interrupt Vector Reg */
#define cprh (( unsigned char*) tmr+6)       /* Preload Hi Reg             */
#define cprm (( unsigned char*) tmr+8)       /* Preload Mid Reg            */
#define cprl (( unsigned char*) tmr+10)      /* Preload Lo Reg             */
#define cnrh (( unsigned char*) tmr+12)      /* Counter Hi Reg             */
#define cnrm (( unsigned char*) tmr+14)      /* Counter Mid Reg            */
#define cnrl (( unsigned char*) tmr+16)      /* Counter Lo Reg             */
#define tsr  (( unsigned char*) tmr+20)      /* Timer Status Reg           */
#define tvector 0x40                         /* Timer Vector reg           */
#define tmrcntrl 0x80                        /* Timer Mode                 */

/* Set PI/T */

#define PGCR  ( unsigned char*)0xFE8001      /* PI/T General Control Reg   */
#define PSRR  ( unsigned char*)0xFE8003      /* PI/T Service Routine Reg   */
#define PIVR  ( unsigned char*)0xFE800B      /* PI/T Interrupt Vector Reg  */
#define PSR   ( unsigned char*)0xFE801B      /* PI/T Status Reg            */
#define PACR  ( unsigned char*)0xFE800D      /* PI/T Port A Control Reg    */
#define PADDR ( unsigned char*)0xFE8005      /* Port A Data Direction Reg  */
#define PADR  ( unsigned char*)0xFE8011      /* Port A Data Reg            */
#define PBCR  ( unsigned char*)0xFE800F      /* Port B Control Reg         */
#define PBDDR ( unsigned char*)0xFE8007      /* Port B Data Direction Reg  */
#define PBDR  ( unsigned char*)0xFE8013      /* Port B Data Reg            */

/* All of the above could have been saved in a header file and be included
via #include command */

void isr();
```

```
main ()
{
  long *vtable;
  int count=1250000;

/* set up interrupt mask in status register
   (see p. 21 of the textbook) */

  asm("    move.w  #$2400,SR");

/* set up parallel ports (see p. 657 of the textbook) */

  *PGCR = 0x0F;         /* Disable Port A & B */
  *PADDR = 0x00;        /* Set Port A as input */
  *PBDDR = 0xFF;        /* set Port B as Output  */
  *PSRR = 0x00;         /* set PI/T for no Interrupts */
  *PBCR = 0x00;         /* Set Port B Control */
  *PACR = 0x40;         /* Set Port A Mode */

/****Prepare CPU for an interrupt processing**/

  *tivr = 70;  /* interrupt vector number (see p. 448 of the textbook) */
  vtable = (long *) (70*4);
  /* vtable is an address, so the type of 70*4 which
     normally is a constant is changed to an address type by the
     casting operator (long *)                                          **/

  *vtable = (long) isr;

/****Set up timer control register  (textbook p. 671) */

  *tcr = 0xA0; /* Set Timer Mode: count down to zero and then interrupt */

  /* Set up the initial value of timer. */
  *cprl = (unsigned char) count;
  count = count >> 8; /* shift right 8 bits */
  *cprm = (unsigned char) count;
  count = count >> 8; /* shift right 8 bits */
  *cprh = (unsigned char) count;

  *tcr = 0xA1;   /* Start timer */

  while (1) { /* Create an infinite loop which does nothing*/ }
}


void isr()
{
  printf("Five secs have passed\n\r");

  *PBDR = *PADR;  /* This is really the main job of isr *
                     It copies the content of porta data regsiter (our input
                     port) and then places it to port B (our output port)*/

  *tsr = 0x01; /* reset the ZDS bit so to re-enable the timer */

  asm(" rte"); /* return from exception */
  /* It is to avoid to use the "rts" (return from subroutine) that will be
     automatically attached by the compiler. */
  /* When an interrupt is invoked, the status register which contains the
     interrupt mask is pushed into the stack.  That push operation is not
     executed for subroutine call.  Therefore "rte" is different from
     "rts" in the extra stack pop operation. */
}
```