Mosaic Industries

Software Pulse Counter and Input Port Scanner

APPLICATION NOTE MI-AN-043

Summary

This program periodically (say, every 50 msec) reads a byte-wide input port. It performs an exclusive OR of the new port contents with the prior port contents and saves the result along with the new contents in variables to allow a foreground task or program to determine which bits have recently changed state. The state changes are not latched, so the foreground program must look at them more frequenly than the update time of the interrupt (50 msec in this example). Each port bit has an associated 16-bit pulse counter that is incremented each time the port bit changes from a low state to a high state.

Implementation details:

The output compare 4 (OC4) function generates a periodic interrupt. The free running counter rolls over every 131.1 msec, and this sets an upper bound on the period at which this pulse scanner operates. This example configures the OC4 interrupt to occur every 50 ms, but you can easily vary this anywhere from 1 msec to 131 msec (and higher with a simple software change). The OC4 output pin, PA4, is not affected by the interrupt.

```
HEX
10 WIDTH !          \ avoid non-unique names
4 USE.PAGE          \ you can remove this if your memory map is already set up!

\ define the relevant control registers
800E  CONSTANT    TCNT                    \ Timer counter register
801C  CONSTANT    TOC4                    \ Output compare 4 register
8022  CONSTANT    TMSK1                   \ Timer interrupt mask register
8023  CONSTANT    TFLG1                   \ Timer interrupt flag register

10    CONSTANT    OC4.MASK                \ Isolates OC4 interrupt flag & mask bits

DECIMAL      \ for clarity, define the period in decimal base
500          CONSTANT TCNTS/MSEC          \ Number of 2ms counts per millisecond

50 TCNTS/MSEC *   CONSTANT    PERIOD.CNT  \ Number of TCNTs in the scan period

HEX
\ By changing the PERIOD.CNT, the time interval can easily
\ be changed.  The PERIOD.CNT, should not be greater than 65535 (131ms)

\ keep these next 2 variable declarations together!
\ We also assume that variables are in common memory (which is true
\ for the default memory maps set by USE.PAGE, etc.)
\ If desired, these two variables can be fetched with 1 uninterrupted
\ |2@| operation from a foreground routine to ensure that an interrupt
\ does not occur between the examination of the two of them:
VARIABLE     INPUT.PORT.STATE  \ access via C@ and C!
VARIABLE     CHANGED.PORT.BITS \ access via C@ and C!;
             \ = (prior.state XOR current.state), which equals 1 for each bit that
             \ has changed since the prior read.
VARIABLE     PULSE.COUNTER.BASE      \ array of 8 2-byte counters
E VALLOT            \ allot a total of 16 bytes (variable alloted 2 bytes)
```

```
: PULSE.COUNTER   ( bit.index -- pulse.counter.address )
\     7 UMIN                                    \ optional clamp of input parameter
      2* PULSE.COUNTER.BASE ROT XN+
      ;


\ for this example, we assume that PORTA is the input port to be scanned.
\ You can use any port you want:
\     PPA, PORTE, input port on the QED Digital I/O Board, etc.

CODE OC4.SERVICE  ( -- )
\ interrupt service routine, assembly coded for optimum speed
      OC4.MASK IMM LDAB
      TFLG1 EXT STAB                  \ clear the interrupt flag that got us here
      TOC4 EXT LDD
      PERIOD.CNT IMM ADDD
      TOC4 EXT STD                    \ set the next time for interrupt to occur
      >FORTH
            PORTA C@                  \ you can specify any input port you want here
      >ASSM                                 ( new.contents -- )
      INPUT.PORT.STATE DROP EXT LDAB        \ A <- prior.contents
      1 IND,Y EORB                          \ B <- prior.contents XOR new.contents
      CHANGED.PORT.BITS DROP EXT STAB       \ save changed bits in variable
      1 IND,Y LDAA                          \ A <- new.contents
      INY INY                               ( -- ) \ clear data stack
      INPUT.PORT.STATE DROP EXT STAA        \ save new.contents in variable
      CHANGED.PORT.BITS DROP EXT ANDA       \ A <- bits.changed.AND.now=1: rising edge
      PULSE.COUNTER.BASE DROP IMM LDX       \ X <- base of counter array
      BEGIN,
            TAB                             \ make a copy of rising edge indicator
            1 IMM ANDB                      \ test lsbit
            NE IF,
                  1 IND,X LDAB
                  INCB
                  1 IND,X STAB
                  EQ IF,                    \ handle rollover
                        0 IND,X LDAB
                        INCB
                        0 IND,X STAB
                  ENDIF,
            ENDIF,
            INX INX                         \ X points to next pulse counter
            LSRA                            \ shift down 1 bit...
      EQ UNTIL,                             \ ...until no more rising edges present
      RTS
END.CODE
```

```
\ here's a high level version, presented to show the algorithm:
: HIGH.LEVEL.OC4.SERVICE      ( -- )
      OC4.MASK TFLG1 (C!)            \ clear the interrupt flag that got us here
      TOC4 (@) PERIOD.CNT + TOC4 (!)       \ set the next time for interrupt to occur
      PORTA C@                       \ you can specify any input port you want here
      INPUT.PORT.STATE C@            ( new.contents\prior.contents -- )
      OVER XOR                       ( new.contents\changes -- )
      2DUP CHANGED.PORT.BITS C!      \ save exclusive or
      INPUT.PORT.STATE C!            ( new.contents\changes -- ) \ save new state
      AND                            ( bits.changed.AND.now=1-- ) \ rising edges!
      8 0
      DO    DUP 1 AND                      \ if rising edge occurred...
            IF    1 I PULSE.COUNTER +!     \ increment pulse counter
            ENDIF
            U2/                            \ shift down 1 bit
      LOOP
      DROP                                 \ drop rising edge indicator
;

: DISABLE.OC4      ( -- )
      \ locally disables the interrupt
      OC4.MASK TMSK1 (CLEAR.BITS)
;

: INIT.VARIABLES  ( -- )
      INPUT.PORT.STATE OFF
      CHANGED.PORT.BITS OFF
      PULSE.COUNTER.BASE 10 ERASE
;

\ INIT.OC4 installs the interrupt handler, initializes the timer set point
\ and enables OC4 interrupt mask.
: INIT.OC4 ( -- )
      DISABLE.OC4                          \ stop the interrupt if it's enabled
      INIT.VARIABLES
      TCNT (@) PERIOD.CNT + TOC4 (!)       \ set time for next interrupt to occur
      CFA.FOR OC4.SERVICE OC4.ID ATTACH    \ install interrupt routine OC4.SERVICE
      OC4.MASK TFLG1 (C!)                  \ resets the OC4 interrupt flag
      OC4.MASK TMSK1 (SET.BITS)            \ enables the OC4I interrupt flag
      ENABLE.INTERRUPTS                    \ globally enable interrupts
;

: SHOW      ( -- )      \ lets you check the current status
      CR    ." Current state = " INPUT.PORT.STATE C@ .
            ."  ; Changed bits = " CHANGED.PORT.BITS C@ .
      CR ." Counters are as follows: "
      PULSE.COUNTER.BASE 10 DUMP
;
```


Mosaic Industries
5437 Central Ave Suite 1, Newark, CA 94560 Telephone: (510) 790-8222 Fax: (510) 790-0925