Single wire connects microcontrollers

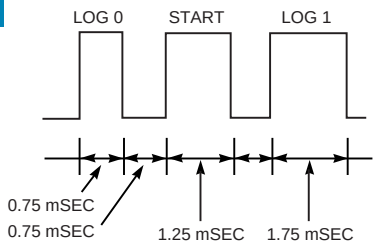
ABEL RAYNUS, ARMITRON INTERNATIONAL, MELROSE, MA

Low-cost µCs, such as Motorola's 68HC705 Series, offer great simplicity at the expense of some useful functions—notably, serial data transmission. Unlike their predecessors, these µCs do not have serial communication interfaces (SCIs), serial peripheral interfaces (SPIs), or simple serial I/O ports (SIOPs). This method describes how you can overcome this deficiency by creating an asynchronous serial interface through µC software. The most obvious way to effect the interface is to use pulse-width coding to differentiate the start pulse and the logic 1 and 0 pulses. You can use any value of pulse-width ratio, depending on your design objectives. This application uses the 1-to-2-to-3 ratio, slightly modified for easy programming. So, logic 0, start, and logic 1 have widths of 0.75, 1.25, and 1.75 msec, respectively (**Figure 1**).

This example uses 68HC705J1A µCs. The information to transmit accumulates in the output data register of the transmitting µC (**Figure 2**). The data-transmission subroutine (**Listing 1** shows a fragment) generates the start pulse, which is followed by an 8-bit data word that reflects the state of the output data register. This pulse sequence goes, LSB-first, to the data-in input of the receiving µC. In the µC transmitter, you can use any output pin as data out. In the µC receiver, you can use any one of the four lower PortA pins (PA0 through PA3) as data in.

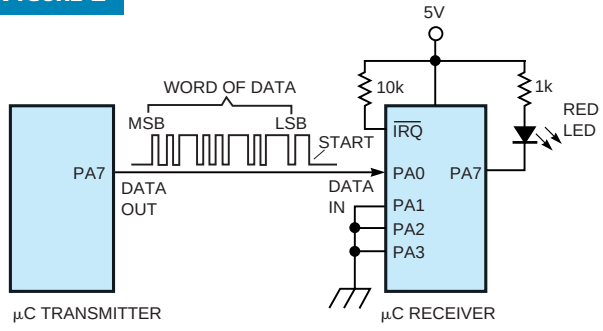
You should program the input pins as positive-edge, external-interrupt inputs. Because pins PA0 to PA3 combine in a logic-OR operation in the µC, you should connect the unused pins to ground to avoid false interruption. You should disable the $\overline{IRQ}$ pin by connecting it to 5V. The external-interrupt subroutine (**Listing 2**) restores the data word, which can generate the proper response according to your design objectives. **Listing 3** shows a fragment of the receiver routine. The program's watchdog utility lights a red LED to indicate that the communication link between the µCs is broken or that it received the wrong sequence. The method also applies to

FIGURE 1



Pulse-width ratios provide a convenient way to transmit serial data between µCs lacking communications amenities.

FIGURE 2



With the help of some µC software, a simple one-wire connection provides serial communications between low-cost µCs.

LISTING 1—TRANSMITTER-PROGRAM FRAGMENT

```
                    1  ************************************
                    2  *  TRANSMITTER PROGRAM FRAGMENT    *
                    3  ************************************
                    4  * Generates the sequence of start pulse and 8-bit word
                    5  * of data according to the content of register REG.
                    6  ****************************************************
                    7  *nclist
0000                8  $include "std-j1a.asm"
                    9  *list
                   10  *  I/O PORTS
0000               11  data_out  equ  7  ;prtA
                   12  *  VARIABLES
00C0               13          org     RAM
00C0               14  REG     rmb     1 ;output data register
00C1               15  num     rmb     1 ;bit test register
                   16  *  INITIALIZATION
07F1               17          org  MOR
07F1 00            18          fcb  $00
0300               19          org  ROM
0300 9C            20  init    rsp     ;reset stack pointer to $ff
0301 A680          21          lda     #$80 ; pA7 as output
0303 B704          22          sta     ddrA
0305 3F00          23          clr     prtA
0307 3FC0          24          clr     REG
0309 3FC1          25          clr     num
                   26  *  DATA TRANSMISSION SUBROUTINE
030B AE7D          27          ldx     #125T   ;start pulse (1.25ms)
030D CD0335        28          jsr     pulse
0310 A601          29          lda     #$01    ;0-bit test prepare
0312 B7C1          30          sta     num
0314 B6C0          31  w1      lda     REG     ;is tested bit = 0?
0316 B4C1          32          and     num
0318 270B          33          beq     w2
031A AEAF          34          ldx     #175T   ;logic1 pulse (1.75ms)
031C CD0335        35          jsr     pulse
031F 98            36  w3      clc             ;0 -> C-carry bit
0320 38C1          37          lsl     num     ;go to next tested bit
0322 24F0          38          bcc     w1      ;is it NOT a last bit?
0324 81            39          rts     ;return from DATA TRANSMISSION
0325 AE4B          40  w2      ldx     #75T    ;log0 pulse (0.75ms)
0327 CD0335        41          jsr     pulse
032A 20F3          42          bra     w3
                   43  ************************************************
032C A602          44  dly01x  lda     #2      ;Delay =0.01*x [ms]
032E 4A            45  rep0    decA
032F 26FD          46          bne     rep0
0331 5A            47          decx
0332 26F8          48          bne     dly01x
0334 81            49          rts             ;return from dly01x
                   50  ************************************************
0335 1E00          51  pulse   bset    data_out,prtA     ; width = x
0337 CD032C        52          jsr     dly01x
033A 1F00          53          bclr    data_out,prtA
033C AE40          54          ldx     #64T
033E CD032C        55          jsr     dly01x
0341 81            56          rts             ;return from pulse
                   57  ************************************************
0342 80            58  un      rti     ;return from unused interrupts
07F8               59          org     VECTORS
07F8 0342          60          fdb     un      ;Timer Interrupt unused
07FA 0342          61          fdb     un      ;External Interrupt unused
07FC 0342          62          fdb     un      ;SWI unused
07FE 0300          63          fdb     init    ;set restart address
```

wireless applications with minor modifications. You can download the complete **listings** from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the files from DI-SIG, #2265. (DI #2265). EDN

To Vote For This Design, Circle No. 349

LISTING 2—EXTERNAL-INTERRUPT SUBROUTINE

```
ART10.ASM              Assembled with IASM   04/09/1998  11:26  PAGE 2



                    57  * EXTERNAL INTERRUPT SUBROUTINE
 032E 3FC1          58  ExtInt: clr     T
 0330 3FC0          59          clr     W
 0332 000011        60  e0      brset   data,prtA,e1 ;High level ?
 0335 01C31C        61          brclr   WF,flag,e2   ;WF=0?
 0338 B6C0          62          lda     W
 033A A101          63          cmp     #1           ;W=1?
 033C 2724          64          beq     log0
 033E A103          65          cmp     #3           ;W=3?
 0340 2728          66          beq     log1
 0342 11C3          67          bclr    WF,flag      ;0 -> WF
 0344 2041          68          bra     e3
 0346 3CC1          69  e1      inc     T
 0348 B6C1          70          lda     T
 034A A137          71          cmp     #55T
 034C 26E4          72          bne     e0
 034E 3FC1          73          clr     T
 0350 3CC0          74          inc     W
 0352 20DE          75          bra     e0
 0354 B6C0          76  e2      lda     W
 0356 A102          77          cmp     #2           ;W=2?
 0358 262D          78          bne     e3
 035A 10C3          79          bset    WF,flag      ;1 -> WF
 035C A6FE          80          lda     #$fe         ;0 -> 0-bit of num
 035E B7C4          81          sta     num
 0360 2025          82          bra     e3
 0362 B6C5          83  log0    lda     reg          ;put 0 into given bit of
 0364 B4C4          84          and     num          ;reg. without changing
 0366 B7C5          85          sta     reg          ;of the rest of its bits.
 0368 200C          86          bra     e4
 036A B6C4          87  log1    lda     num          ;put 1 into given bit of
 036C B7C6          88          sta     mem          ;reg. without changing
 036E B4C5          89          and     reg          ;of the rest of its bits
 0370 33C6          90          com     mem
 0372 B8C6          91          eor     mem
 0374 B7C5          92          sta     reg
 0376 99            93  e4      sec                  ;1 -> Carry bit
 0377 39C4          94          rol     num          ;go to the next bit
 0379 250C          95          bcs     e3           ;is it NOT the last bit?
 037B 11C3          96          bclr    WF,flag      ;0 -> WF word process flag
 037D 17C3          97          bclr    WDF,flag
;0+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

LISTING 3—RECEIVER-PROGRAM FRAGMENT

```
                     1
                     2  *     RECEIVER  PROGRAM FRAGMENT
                     3  ********************************
                     4  * Transfers the received serial data
                     5  * into content of the register WORD.
                     6  ****************************************************
                     7  *nolist
0000                 8  $include "std-jla.asm"
                     9  *list
07F1                10          org  MOR  ;pos.edge Ext.Interrupt
07F1 24             11          fcb  $24  ;  on pA0 - pA3 enable
                    12  *I/O PORTS
07F2                13  data    equ  0  ;prtA Data Input pin
07F2                14  RedLED equ  7  ;prtA red LED output pin
                    15  *Specific equates
07F2                16  WF      equ  0  ;Word processing flag
07F2                17  F1      equ  1  ;signal presense flag
07F2                18  WDF     equ  3  ;watch-dog flag
                    19  * VARIABLES
00C0                20          org     RAM
00C0                21  W       rmb     1 ;pulse width counter
00C1                22  T       rmb     1 ;time (0.5 ms) counter
00C2                23  wdc     rmb     1 ;watch-dog counter
00C3                24  flag    rmb     1 ;flag register
00C4                25  num     rmb     1 ;register to form word
00C5                26  reg     rmb     1 ;temporary word register.
00C6                27  mem     rmb     1 ;memory register
00C7                28  WORD    rmb     1 ;final received word register.
                    29  * INITIALIZATION
0300                30          org     ROM
0300 9C             31  init    rsp             ;reset stack pointer to $ff
0301 A6F0           32          lda     #$f0        ;pA0 - pA3 as input
0303 B704           33          sta     ddrA        ;pA4 - pA7 as output
0305 CD0318         34          jsr     in_set      ;go to initial set
0308 1E0A           35          bset    IRQE,ISCR   ;ExtInt enable
030A 1A08           36          bset    TOIE,TSCR   ;TOF interrupt enable
030C 9A             37          cli                 ;interrupt enable
030D 06C303         38  m0      brset   WDF,flag,m1 ;WDF=1? No data-in?
0310 00C3FA         39          brset   WF,flag,m0  ;WF=1? wait for word end
0313 CD0318         40  m1      jsr     in_set
0316 20F5           41          bra     m0
                    42  ****************************************************
0318 3F00           43  in_set  clr     prtA        ;set red LED on
031A 5F             44          clrx                ;start to clear
031B 6FC0           45  a0      clr     RAM,x       ; 8 variables in RAM
031D 5C             46          incx
031E A308           47          cpx     #8T
0320 25F9           48          blo     a0
0322 81             49          rts                 ;return from in_set
                    50  ****************************************************
0323 3CC2           51  TOFint  inc     wdc
0325 3DC2           52          tst     wdc         ;wdc / 0 ?
0327 2602           53          bne     t0
0329 16C3           54          bset    WDF,flag    ;1 -> WDF
032B 1608           55  t0      bset    TOFR,TSCR   ;TOF reset
032D 80             56          rti                 ;return from TOFint
```