intel®

AB-41

APPLICATION
BRIEF

Software Serial Port Implemented with the PCA

BETSY JONES
ECO APPLICATIONS ENGINEER

August 1988

Order Number: 270531-002

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

SOFTWARE SERIAL PORT IMPLEMENTED WITH THE PCA

CONTENTS

For microcontroller applications which require more than one serial port, the 83C51FA Programmable Counter Array (PCA) can implement additional half-duplex serial ports. If the on-chip UART is being used as an inter-processor link, the PCA can be used to interface the 83C51FA to additional asynchronous lines.

This application uses several different Compare/Capture modes available on the PCA to receive or transmit bytes of data. It is assumed the reader is familiar the PCA and ASM51. For more information on the PCA refer to the "Hardware Description of the 83C51FA" chapter in the Embedded Controller Handbook (Order No. 210918).

Introduction

The figure below shows the format of a standard 10-bit asynchronous frame: 1 start bit (0), 8 data bits, and 1 stop bit (1). The start bit is used to synchronize the receiver to the transmitter; at the leading edge of the start bit the receiver must set up its timing logic to sample the incoming line in the center of each bit. Following the start bit are eight data bits which are transmitted least significant bit first. The stop bit is set to the opposite state of the start bit to guarantee that the leading edge of the start bit will cause a transition on the line. It also provides a dead time on the line so that the receiver can maintain its synchronization.

Two of the Compare/Capture modes on the PCA are used in receiving and transmitting data bits. When receiving, the Negative-Edge Capture mode allows the PCA to detect the start bit. Then using the Software Timer mode, interrupts are generated to sample the incoming data bits. This same mode is used to clock out bits when transmitting.

This Application Note contains four sections of code:

(1) List of variables
(2) Initialization routine
(3) Receive routine
(4) Transmit routine.

A complete listing of the routines and the test loop which was used to verify their operation is found in the Appendix. A total of three half-duplex channels were run at 2400 Baud in the test program. The listings shown here are simplified to one channel (Channel 0).

Variables

Listing 1 shows the variables used in both the receive and transmit routines. Flags are defined to signify the status of the reception or transmission of a byte (e.g. RCV__START__BIT, TXM__START__BIT). RCV__BUF and TXM__BUF simulate the on-chip serial port SBUF as two separate buffer registers. The temporary registers, RCV__REG and TXM__REG, are used to save bits as they are received or transmitted. Finally, two counter registers keep track of how many bits have been received or transmitted.

Variables are also needed to define one-half and one-full bit times in units of PCA timer ticks. (One bit time = 1 / baud rate.) With the PCA timer incremented every machine cycle, the equation to calculate one bit time can be written as:

$$\frac{\text{Osc. Freq.}}{(12) \times (\text{baud rate})} = 1 \text{ bit time (in PCA timer ticks)}$$

In this example, the baud rate is 2400 at 16 MHz.

$$\frac{16 \text{ MHz}}{(12) \times (2400)} = 556 \text{ counts} = 22\text{C Hex}$$

The high and low byte of this value is placed in the variables FULL__BIT__HIGH and FULL__BIT__LOW, respectively. 115H is the value loaded into HALF__BIT__HIGH and HALF__BIT__LOW.

270531-1

Listing 1. Variables used by the software serial port. Channel 0

```
;
;  Receive Routine
;
RCV_START_BIT_0   BIT       20H.0     ; Indicates  start bit
                                      ; has been received
RCV_DONE_0        BIT       20H.1     ; Indicates  data byte
                                      ; has been received
RCV_BUF_0         DATA      30H       ; Software Receive
                                      ; "SBUF"
RCV_REG_0         DATA      31H       ; Temporary register
                                      ; for receive bits
RCV_COUNT_0       DATA      32H       ; Counter for receiving
                                      ; bits
;  Transmit Routine:
;
TXM_START_BIT_0   BIT       20H.3     ; Indicates start bit
                                      ; has been transmitted
TXM_IN_PROGRESS_0 BIT       20H.4     ; Indicates transmit is
                                      ; in progress
TXM_BUF_0         DATA      34H       ; Software transmit
                                      ; "SBUF"
TXM_REG_0         DATA      35H       ; Temporary register
                                      ; for transmitting bits
TXM_COUNT_0       DATA      36H       ; Counter for transmit-
                                      ; ting bits
DATA_0            DATA      37H       ; Register used for the
                                      ; test program
;
NEG_EDGE          EQU       11H       ; Two modes of operation
S_W_TIMER         EQU       49H       ; for compare/capture
                                      ; modules
;
HALF_BIT_HIGH     EQU       01H       ; Half bit time = 115H
HALF_BIT_LOW      EQU       15H
FULL_BIT_HIGH     EQU       02H       ; Full bit time = 22CH
FULL_BIT_LOW      EQU       2CH       ; 2400 Baud at 16 MHz
```

270531-4

Initialization

Listing 2 contains the intialization code for the receive and transmit process. Module 0 of the PCA is used as a receiver and is first set up to detect a negative edge from the start bit. Modules 2 and 3 are used for the additional 2 channels (see the Appendix). Module 3 is used as a separate software timer to transmit bits.

Listing 2. Initialization Routine

```
ORG 0000H
LJMP INITIALIZE
ORG 001BH
LJMP RECEIVE_DONE               ; Timer 1 overflow -
                                ; simulates "RI" interrupt
ORG 0033H
LJMP RECEIVE                    ; PCA interrupt
;
INITIALIZE: MOV SP, #5FH        ; Initialize stack pointer
                                ; (specific to test program)
INIT_PCA: MOV CMOD, #00H        ; Increment PCA timer
                                ; @ 1/12 Osc Frequency
          MOV CCON, #00H        ; Clear all status flags
          MOV CCAPM0, #NEG_EDGE ; Module 0 in negative-edge
                                ; trigger mode (P1.3)
          MOV CCAPM3, #S_W_TIMER ; Module 3 as software timer
                                ; mode
          MOV CL, #00H
          MOV CH, #00H
          MOV IE, #0D8H         ; Init all needed interrupts
                                ; EA, EC, ES, ET1
          SETB CR               ; Turn on PCA Counter
```

270531–5

All flags and registers from Listing 1 should be cleared in the initialization process.

Receive Routine

Two operating modes of the PCA are needed to receive bits. The module must first be able to detect the leading edge of a start bit so it is initially set up to capture a 1-to-0 transition (i.e. Negative-Edge Capture mode). The module is then reconfigured as a software timer to cause an interrupt at the center of each bit to deserialize the incoming data. The flowchart for the receive routine is given in Figure 1.

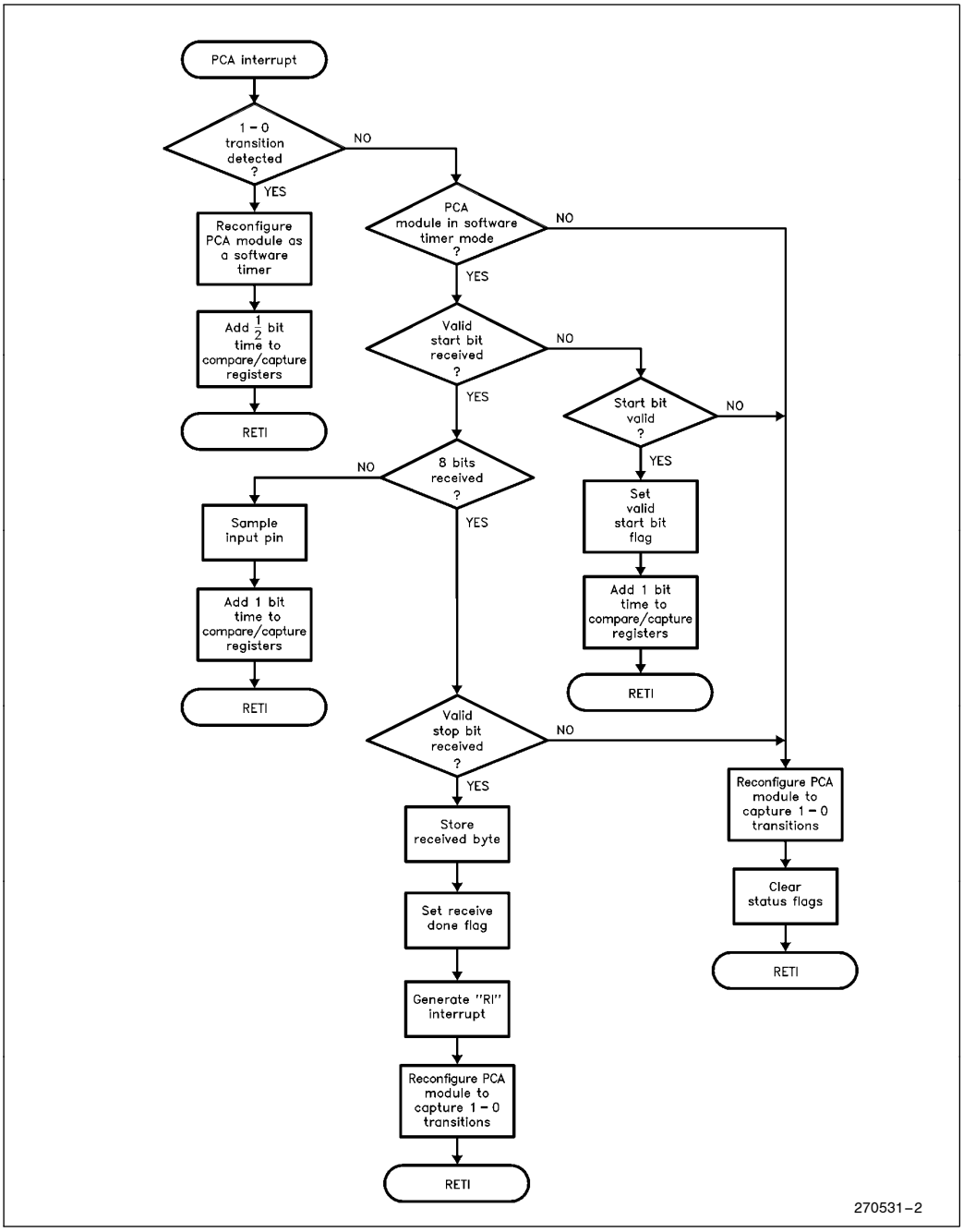

Figure 1. Flowchart for the Receive Routine

Listing 3.1 shows the code needed to detect a start bit. Notice that the first software timer interrupt will occur one-half bit time after the leading edge of the start bit to check its validity. If it is valid, the RCV__START__BIT is set. The rest of the samples will occur a full bit time later. The RCV__COUNT register is loaded with a value of 9 which indicates the number of bits to be sampled: 8 data bits and 1 stop bit.

Listing 3.1. Receive Interrupt Routine

```
RECEIVE:  PUSH ACC
          PUSH PSW
;
MODULE_0: CLR CCF0               ; Assume reception on
                                 ; Module 0
          MOV A, CCAPM0          ; Check mode of module.  If
          ANL A, #01111111B      ; set up to receive negative
          CJNE A,#NEG_EDGE, RCV_START_0 ; edges, then module
                                 ; is waiting for a start  bit
;
          CLR C                  ; Update compare/capture
          MOV A, #HALF_BIT_LOW   ; registers for half bit time
          ADD A, CCAP0L          ; to sample start bit
          MOV CCAP0L, A          ; Half bit time = 115H
          MOV A, #HALF_BIT_HIGH
          ADDC A, CCAP0H
          MOV CCAP0H, A
          MOV CCAPM0, #S_W_TIMER ; Reconfigure module 0 as
          POP PSW                ; a software timer to sample
          POP ACC                ; bits
          RETI
;
RCV_START_0: CJNE A, #S_W_TIMER, ERROR_0 ; Check module is
                                 ; configured as a software
                                 ; timer, otherwise error.
          JB RCV_START_BIT_0, RCV_BYTE_0 ; Check if start bit
                                 ; is received yet.
          JB P1.3, ERROR_0       ; Check that start bit = 0,
                                 ; otherwise error.
          SETB RCV_START_BIT_0   ; Signify valid start bit
                                 ; was received
          MOV RCV_COUNT_0, #09H  ; Start counting bits sampled
;
          CLR C                  ; Update compare/capture
          MOV A, #FULL_BIT_LOW   ; registers to sample
          ADD A, CCAP0L          ; incoming bits
          MOV CCAP0L, A          ; Full bit time = 22CH
          MOV A, #FULL_BIT_HIGH
          ADDC A, CCAP0H
          MOV CCAP0H, A
          POP PSW
          POP ACC
          RETI
```

270531-6

The next 8 timer interrupts will receive the incoming data bits; the RCV__COUNT register keeps track of how many bits have been sampled. As each bit is sampled, it is shifted through the Carry Flag and saved in RCV__REG. The ninth sample checks the validity of the stop bit. If it is valid, the data byte is moved into RCV__BUF.

The main routine must have a way to know that a byte has been received. With the on-chip UART, the RI (Receive Interrupt) bit is set whenever a byte has been received. For the software serial port, any unimplemented interrupt vector can be used to generate an interrupt when a byte has been received. This routine uses the Timer 1 Overflow interrupt (its selection is arbitrary). A routine to test this interrupt is included in the listing in the Appendix.

Listing 3.2. Receive Interrupt Routine (Continued)

```
RCV_BYTE_0: DJNZ RCV_COUNT_0, RCV_DATA_0  ; On 9th sample,
                                          ; check for valid stop bit
RCV_STOP_0: JNB P1.3, ERROR_0
          MOV RCV_BUF_0, RCV_REG_0 ; Save received byte in
                                   ; receive "SBUF"
          SETB RCV_DONE_0          ; Flag which module received
                                   ; a byte
          SETB TF1                 ; Generate an interrupt so
                                   ; main program knows a byte
                                   ; has been received
                                   ; (Note: selection of TF1 is
                                   ; arbitrary)
          MOV CCAPM0, #NEG_EDGE    ; Reconfigure module 0 for
                                   ; Reception of a start bit
          POP PSW
          POP ACC
          RETI
;
RCV_DATA_0: MOV C, P1.3            ; Sampling data bits
          MOV A, RCV_REG_0         ; Shifts bits thru CY into
          RRC A                    ; ACC
          MOV RCV_REG_0, A         ; Save each reception in
                                   ; temporary register
          CLR C                    ; Update c/c register for
          MOV A, #FULL_BIT_LOW     ; next sample time
          ADD A, CCAP0L
          MOV CCAP0L, A
          MOV A, #FULL_BIT_HIGH
          ADDC A, CCAP0H
          MOV CCAP0H, A
          POP PSW
          POP ACC
          RETI
```

270531-7

In addition, an error routine (Listing 3.3) is included for invalid start or stop bits to offer some protection against noise. If an error occurs, the module is re-initialized to look for another start bit.

Listing 3.3 Error Routine for Receive Routine

```
ERROR_0:  MOV CCAPM0, #NEG_EDGE  ; Reset module to look for
                                 ; start bit
          CLR RCV_START_BIT_0    ; Clear flags which might
                                 ; have been set
          POP PSW
          POP ACC
          RETI
```

270531-8

Transmit Routine

Another PCA module is configured as a software timer to interrupt the CPU every bit time. With each timer interrupt one or more bits can be transmitted through port pins. In the test program three channels were operated simultaneously, but in the listings below, one channel is shown for simplicity. The selection of port pins is user programmable. The flowchart for the transmit routine is given in Figure 2.

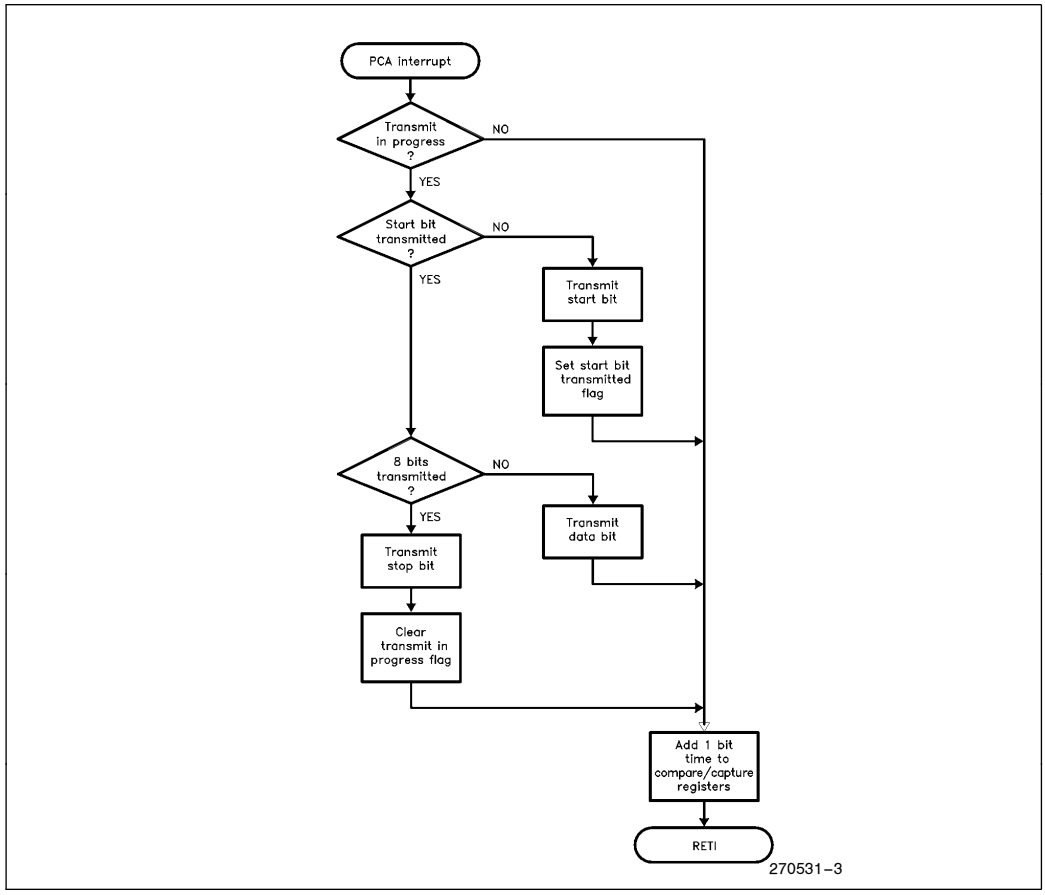

Figure 2. Flowchart for the Transmit Routine

When a byte is ready to be transmitted, the main program moves the data byte into the TXM__BUF register and sets the corresponding TXM__IN__PROGRESS bit. This bit informs the interrupt routine which channel is transmitting. The data byte is then moved in the storage register TXM__REG, and the TXM__COUNT is loaded. This main routine is shown in Listing 4.1.

Listing 4.1 Transmit Set Up Routine. Channel 0.

```
TXM_ON_0: CLR TXM_START_BIT_0    ; Clear status flag from
                                 ; previous transmission
          MOV TXM_BUF_0, DATA_0  ; Load "SBUF" with data byte
          MOV TXM_REG_0, TXM_BUF_0
          MOV TXM_COUNT_0, #09   ; 8 data bits + 1 stop bit
          SETB TXM_IN_PROGRESS_0
```

270531–9

Listing 4.2 shows the transmit interrupt routine. The first time through, the start bit is transmitted. As each successive interrupt outputs a bit, the contents of TXM__REG is shifted right one place into the Carry flag, and the TXM__COUNT is decremented. When TXM__COUNT equals zero, the stop bit is transmitted.

Listing 4.2. Transmit Interrupt Routine

```
TRANSMIT: PUSH ACC
          PUSH PSW
          CLR CCF3                ; Clear s/w timer interrupt
                                  ; for transmitting bits
          JNB TXM_IN_PROGRESS_0, TRANSMIT_1 ; Check which
                                  ; channel is transmitting.
                                  ; "TRANSMIT_1" is listed in
                                  ; the Appendix
;
TRANSMIT_0: JB TXM_START_BIT_0, TXM_BYTE_0  ; If start bit
                                  ; has been sent, continue
                                  ; transmitting bits.
          CLR P3.2                ; Otherwise transmit start
                                  ; bit
          SETB TXM_START_BIT_0    ; Signify start bit sent
          JMP TXM_EXIT
;
TXM_BYTE_0: DJNZ TXM_COUNT_0, TXM_DATA_0  ; If bit count
                                  ; equals 1 thru 9, transmit
                                  ; data bits (8 total)
;
TXM_STOP_0: SETB P3.2             ; When bit count = 0,
                                  ; transmit stop bit
          CLR TXM_IN_PROGRESS_0   ; Indicate transmission is
                                  ; finished and ready for
                                  ; next byte
          JMP TXM_EXIT
;
TXM_DATA_0: MOV A, TXM_REG_0      ; Transmit one bit at a time
          RRC A                   ; through the carry bit
          MOV P3.2, C
          MOV TXM_REG_0, A        ; Save what's not been sent
;
TXM_EXIT: CLR C                   ; Update compare value with
          MOV A, #FULL_BIT_LOW    ; Full bit time = 22CH
          ADD A, CCAP3L
          MOV CCAP3L, A
          MOV A, #FULL_BIT_HIGH
          ADDC A, CCAP3H
          MOV CCAP3H, A
          POP PSW
          POP ACC
          RETI
```

270531-10

Conclusion

The software routines in the Appendix can be altered to vary the baud rate and number of channels to fit a particular application. The number of channels which can be implemented is limited by the CPU time required to service the PCA interrupt. At higher baud rates, fewer channels can be run.

The test program verifies the simultaneous operation of three half-duplex channels at 2400 Baud and the on-chip full-duplex channel at 9600 Baud. Thirty-three percent of the CPU time is required to operate all four channels. The test was run for several hours with no apparent malfunctions.

```
MCS-51 MACRO ASSEMBLER    SWPORT                                    01/01/80   PAGE    1


DOS 3.20 (038-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN SWPORT.OBJ
ASSEMBLER INVOKED BY:  C:\AEDIT\ASM51.EXE SWPORT.RCV


LOC  OBJ            LINE     SOURCE

                       1     $NOMOD51
                       2     $NOSYMBOLS
                       3     $NOLIST
                     152     ;
                     153     ;
                     154     ;
                     155     ; This program tests the receive routines of a software serial port.
                     156     ; Three half-duplex channels are implemented in software to run at
                     157     ; 2400 Baud (16MHz).  The on-chip serial port is also running full-duplex
                     158     ; at 9600 Baud.  Thirty-three percent of the CPU time is required to run
                     159     ; all four ports simultaneously.
                     160     ;
                     161     ; To test the receive routines, "dummy" terminals transmit 00 - FF hex
                     162     ; continually to the PCA.  When the first byte is received, it is
                     163     ; compared with 00.  If the comparison is valid, the compare value is
                     164     ; incremented and the routine waits to receive the next byte.  Error
                     165     ; routines toggle various Port 3 pins if an invalid comparison occurs
                     166     ; or if an invalid start bit or stop bit is received.
                     167     ;
0000                 168     ORG 00H
0000 020036          169     LJMP INITIALIZE
                     170     ;
001B                 171     ORG 001BH
001B 02025C          172     LJMP RECEIVE_DONE              ; Timer 1 Overflow - simulates "RI" interrupt
                     173     ;
0023                 174     ORG 0023H
0023 020282          175     LJMP SERIAL_PORT               ; Serial port interrupt
                     176     ;
0033                 177     ORG 0033H                      ; PCA interrupt
0033 0200DC          178     LJMP RECEIVE
                     179     ;
                     180     ;
                     181     ;               VARIABLES USED BY THE SOFTWARE SERIAL PORT
                     182     ;               =========================================
                     183     ;
                     184     ;  RECEIVE ROUTINE:
                     185     ;  ----------------
                     186     ;
  0000               187     RCV_START_BIT_0         BIT             20H.0   ; Indicates start bit has been
  0008               188     RCV_START_BIT_1         BIT             21H.0   ; received
  0010               189     RCV_START_BIT_2         BIT             22H.0
                     190     ;
  0001               191     RCV_DONE_0              BIT             20H.1   ; Indicates data byte has been
  0009               192     RCV_DONE_1              BIT             21H.1   ; received
  0011               193     RCV_DONE_2              BIT             22H.1
                     194     ;
  0002               195     RCV_ON_0                BIT             20H.2   ; Used in main test program to check
  000A               196     RCV_ON_1                BIT             21H.2   ; for a received byte
  0012               197     RCV_ON_2                BIT             22H.2
                     198     ;
```

270531-11

```
MCS-51 MACRO ASSEMBLER    SWPORT                                        01/01/80   PAGE    2

LOC  OBJ            LINE     SOURCE

  0030              199      RCV_BUF_0             DATA        30H     ; Software receive "SBUF"
  0040              200      RCV_BUF_1             DATA        40H
  0050              201      RCV_BUF_2             DATA        50H
                    202      ;
  0031              203      RCV_REG_0             DATA        31H     ; Temporary register for
  0041              204      RCV_REG_1             DATA        41H     ; receiving bits
  0051              205      RCV_REG_2             DATA        51H
                    206      ;
  0032              207      RCV_COUNT_0           DATA        32H     ; Counter for receiving bits
  0042              208      RCV_COUNT_1           DATA        42H
  0052              209      RCV_COUNT_2           DATA        52H
                    210      ;
  0033              211      COUNT_0               DATA        33H     ; Used in test program to check
  0043              212      COUNT_1               DATA        43H     ; bytes being received
  0053              213      COUNT_2               DATA        53H
                    214      ;
  0011              215      NEG_EDGE              EQU         11H     ; Two modes of operation for the
  0049              216      S_W_TIMER             EQU         49H     ; Compare/Capture modules
                    217      ;
  0015              218      HALF_BIT_LOW          EQU         15H     ; Half bit time = 115H
  0001              219      HALF_BIT_HIGH         EQU         01H
  002C              220      FULL_BIT_LOW          EQU         2CH     ; Full bit time = 22CH
  0002              221      FULL_BIT_HIGH         EQU         02H
                    222                                                ; 2400 Baud @ 16MHz
                    223      ;
                    224      ;
                    225
                    226      ;
                    227      ;                                  INITIALIZATION ROUNTINE
                    228      ;                                  =======================
                    229      ;
0036 75815F         230      INITIALIZE:     MOV SP, #5FH               ; Initialize stack pointer
                    231                                                 ; (specific to the test program)
0039 75D900         232      INIT_PCA:       MOV CMOD, #00H             ; Increment PCA clock @ 1/12 Osc Freq
003C 75D800         233                      MOV CCON, #00H             ; Clear all status flags
003F 75DA11         234                      MOV CCAPM0, #NEG_EDGE      ; Module 0 in Neg-edge capture mode (P1.3)
0042 75DB11         235                      MOV CCAPM1, #NEG_EDGE      ; Module 1         "                (P1.4)
0045 75DC11         236                      MOV CCAPM2, #NEG_EDGE      ; Module 2         "                (P1.5)
                    237      ;
0048 75E900         238                      MOV CL, #00H
004B 75F900         239                      MOV CH, #00H
004E 75A8D8         240                      MOV IE, #0D8H              ; Initialize needed interrupt: EA,EC,ES,ET1
0051 D2DE           241                      SETB CR                    ; Turn on PCA counter
                    242      ;
0053 759850         243      INIT_SP:        MOV SCON, #50H             ; Serial port in mode 1 (8-Bit UART)
0056 75CBFF         244                      MOV RCAP2H, #0FFH          ; Reload values for 9600 Baud @ 16 MHz
0059 75CACC         245                      MOV RCAP2L, #0CCH
005C 75C834         246                      MOV T2CON, #34H            ; Timer 2 as a baud-rate generator,
                    247                                                 ; turn on timer 2
                    248      ;
005F C200           249      INIT_FLAGS:     CLR RCV_START_BIT_0
0061 C208           250                      CLR RCV_START_BIT_1
0063 C210           251                      CLR RCV_START_BIT_2
                    252      ;
0065 C201           253                      CLR RCV_DONE_0
```

270531-12

```
LOC  OBJ            LINE     SOURCE

0067 C209            254                 CLR RCV_DONE_1
0069 C211            255                 CLR RCV_DONE_2
                     256     ;
006B C202            257                 CLR RCV_ON_0
006D C20A            258                 CLR RCV_ON_1
006F C212            259                 CLR RCV_ON_2
                     260     ;
                     261     ; Port 3 pins used in test program for error routines
                     262     ;
                     263     ; Main program:
0071 D2B2            264                 SETB P3.2                   ; Error in comparison on module 0
0073 D2B3            265                 SETB P3.3                   ; Error in comparison on module 1
0075 D2B4            266                 SETB P3.4                   ; Error in comparison on module 2
                     267     ;
                     268     ; Interrupt routines:
0077 D2B5            269                 SETB P3.5                   ; Error in reception on module 0
0079 D2B6            270                 SETB P3.6                   ; Error in reception on module 1
007B D2B7            271                 SETB P3.7                   ; Error in reception on module 2
                     272     ;
007D 753000          273                 MOV RCV_BUF_0, #00H
0080 754000          274                 MOV RCV_BUF_1, #00H
0083 755000          275                 MOV RCV_BUF_2, #00H
                     276     ;
0086 753200          277                 MOV RCV_COUNT_0, #00H
0089 754200          278                 MOV RCV_COUNT_1, #00H
008C 755200          279                 MOV RCV_COUNT_2, #00H
                     280     ;
008F 753100          281                 MOV RCV_REG_0, #00H
0092 754100          282                 MOV RCV_REG_1, #00H
0095 755100          283                 MOV RCV_REG_2, #00H
                     284     ;
0098 753300          285                 MOV COUNT_0, #00H
009B 754300          286                 MOV COUNT_1, #00H
009E 755300          287                 MOV COUNT_2, #00H
                     288     ;
                     289     ;
                     290     ;
                     291     ;                          MAIN TEST ROUTINE - RECEIVE BITS
                     292     ;                          ================================
                     293     ;
00A1 300209          294     CHECK_0:    JNB RCV_ON_0, CHECK_1       ; Main program continually checks
00A4 E530            295                 MOV A, RCV_BUF_0            ; each channel for a received byte.
00A6 B5331E          296                 CJNE A, COUNT_0, ERROR0     ; Once a byte is received, it is compared
00A9 C202            297                 CLR RCV_ON_0                ; with the current value in the "COUNT"
00AB 0533            298                 INC COUNT_0                 ; register
                     299     ;
00AD 300A09          300     CHECK_1:    JNB RCV_ON_1, CHECK_2
00B0 E540            301                 MOV A, RCV_BUF_1
00B2 B54319          302                 CJNE A, COUNT_1, ERROR1
00B5 C20A            303                 CLR RCV_ON_1
00B7 0543            304                 INC COUNT_1
                     305     ;
00B9 3012E5          306     CHECK_2:    JNB RCV_ON_2, CHECK_0
00BC E550            307                 MOV A, RCV_BUF_2
00BE B55314          308                 CJNE A, COUNT_2, ERROR2
```

270531-13

```
MCS-51 MACRO ASSEMBLER    SWPORT                                        01/01/80   PAGE    4

LOC  OBJ            LINE     SOURCE

00C1 C212            309                 CLR RCV_ON_2
00C3 0553            310                 INC COUNT_2
00C5 80DA            311                 JMP CHECK_0
                     312     ;
00C7 C2B2            313     ERROR0:     CLR P3.2                    ; Error in comparison on module 0
00C9 75DA00          314                 MOV CCAPM0, #00H            ; Discontinue receiving bytes
00CC 80DF            315                 JMP CHECK_1
                     316     ;
00CE C2B3            317     ERROR1:     CLR P3.3                    ; Error in comparison on module 1
00D0 75DB00          318                 MOV CCAPM1, #00H
00D3 80E4            319                 JMP CHECK_2
                     320     ;
00D5 C2B4            321     ERROR2:     CLR P3.4                    ; Error in comparison on module 2
00D7 75DC00          322                 MOV CCAPM2, #00H
00DA 80C5            323                 JMP CHECK_0
                     324     ;
                     325     ;
                     326     ;
                     327     ;                       PCA INTERRUPT ROUNTINE - RECEIVE BITS
                     328     ;                       =====================================
                     329     ;
00DC C0E0            330     RECEIVE:    PUSH ACC
00DE C0D0            331                 PUSH PSW
                     332     ;
00E0 20D811          333                 JB CCF0, MODULE_0           ; Check which module caused
00E3 20D908          334                 JB CCF1, JUMP_1             ; PCA interrupt and jump to
00E6 20DA08          335                 JB CCF2, JUMP_2             ; appropriate routine
00E9 D0D0            336                 POP PSW
00EB D0E0            337                 POP ACC
00ED 32              338                 RETI
                     339     ;
00EE 02016C          340     JUMP_1:     LJMP MODULE_1
00F1 0201E4          341     JUMP_2:     LJMP MODULE_2
                     342     ;
                     343     ;
                     344     ;
                     345     ;                       CHANNEL 0
                     346     ;                       ---------
                     347     ;
00F4 C2D8            348     MODULE_0:   CLR CCF0                    ; Reception on module 0
00F6 E5DA            349                 MOV A, CCAPM0               ; Check mode of module. If set up to
00F8 547F            350                 ANL A, #01111111B           ; receive negative edges, then module
00FA B41115          351                 CJNE A, #NEG_EDGE, RCV_START_0  ; is waiting for a start bit
                     352     ;
00FD C3              353                 CLR C                       ; Update Compare/Capture registers for
00FE 7415            354                 MOV A, #HALF_BIT_LOW        ; half a bit time
0100 25EA            355                 ADD A, CCAP0L               ; to sample start bit
0102 F5EA            356                 MOV CCAP0L, A               ; Half bit time = 115H
0104 7401            357                 MOV A, #HALF_BIT_HIGH
0106 35FA            358                 ADDC A, CCAP0H
0108 F5FA            359                 MOV CCAP0H, A
010A 75DA49          360                 MOV CCAPM0, #S_W_TIMER      ; Reconfigure module 0 as
                     361                                             ; a software timer to sample bits
010D D0D0            362                 POP PSW
010F D0E0            363                 POP ACC
```

270531-14

```
LOC  OBJ            LINE     SOURCE

0111 32              364                 RETI
                     365     ;
0112 B4494B          366     RCV_START_0:    CJNE A, #S_W_TIMER, ERROR_0     ; Check module is configured
                     367                                                     ; as a software timer, otherwise error.
0115 20001A          368                 JB RCV_START_BIT_0, RCV_BYTE_0  ; Check if start bit
                     369                                                     ; has been received yet
0118 209345          370                 JB P1.3, ERROR_0                ; Check that start bit = 0,
                     371                                                     ; otherwise error.
011B D200            372                 SETB RCV_START_BIT_0            ; Signify valid start bit
                     373                                                     ; was received
011D 753209          374                 MOV RCV_COUNT_0, #09H           ; Start counting bits sampled
                     375     ;
0120 C3              376                 CLR C                           ; Update C/C registers to sample
0121 742C            377                 MOV A, #FULL_BIT_LOW            ; incoming bits
0123 25EA            378                 ADD A, CCAP0L                   ; Full bit time = 22CH
0125 F5EA            379                 MOV CCAP0L, A
0127 7402            380                 MOV A, #FULL_BIT_HIGH
0129 35FA            381                 ADDC A, CCAP0H
012B F5FA            382                 MOV CCAP0H, A
012D D0D0            383                 POP PSW
012F D0E0            384                 POP ACC
0131 32              385                 RETI
                     386     ;
0132 D53212          387     RCV_BYTE_0:     DJNZ RCV_COUNT_0, RCV_DATA_0    ; On 9th sample, check for
                     388                                                     ; valid stop bit
0135 309328          389     RCV_STOP_0:     JNB P1.3, ERROR_0
0138 853130          390                 MOV RCV_BUF_0, RCV_REG_0        ; Save received byte in receive "SBUF"
013B D201            391                 SETB RCV_DONE_0                 ; Flag which module received a byte
013D D28F            392                 SETB TF1                        ; Generate an interrupt so main program
                     393                                                     ; knows a byte has been received
                     394                                                     ; (NOTE: selection of TF1 is arbitrary)
013F 75DA11          395                 MOV CCAPM0, #NEG_EDGE           ; Reconfigure module 0 for next
0142 D0D0            396                 POP PSW                         ; reception of a start bit
0144 D0E0            397                 POP ACC
0146 32              398                 RETI
                     399     ;
0147 A293            400     RCV_DATA_0:     MOV C, P1.3                     ; Sampling data bits
0149 E531            401                 MOV A, RCV_REG_0                ; Shift bits through CY into ACC
014B 13              402                 RRC A
014C F531            403                 MOV RCV_REG_0, A                ; Save each reception in temporary
                     404                                                     ; register
014E C3              405                 CLR C
014F 742C            406                 MOV A, #FULL_BIT_LOW            ; Update C/C register for next
0151 25EA            407                 ADD A, CCAP0L                   ; sample time
0153 F5EA            408                 MOV CCAP0L, A
0155 7402            409                 MOV A, #FULL_BIT_HIGH
0157 35FA            410                 ADDC A, CCAP0H
0159 F5FA            411                 MOV CCAP0H, A
015B D0D0            412                 POP PSW
015D D0E0            413                 POP ACC
015F 32              414                 RETI
                     415     ;
0160 C2B5            416     ERROR_0:        CLR P3.5                        ; Error routine for invalid start or
                     417                                                     ; stop bit or invalid mode comparison
```

270531-15

```
LOC  OBJ            LINE     SOURCE

                      418                                    ; Port pin used for debug only
0162 75DA11           419                    MOV CCAPM0, #NEG_EDGE   ; Reset module to look for start bit
0165 C200             420                    CLR RCV_START_BIT_0     ; Clear flags which might have been set
0167 D0D0             421                    POP PSW
0169 D0E0             422                    POP ACC
016B 32               423                    RETI
                      424     ;
                      425     ;
                      426     ;                      CHANNEL 1
                      427     ;                      ---------
                      428     ;
016C C2D9             429     MODULE_1:      CLR CCF1                ; Similar to module 0
016E E5DB             430                    MOV A, CCAPM1
0170 547F             431                    ANL A, #01111111B
0172 B41115           432                    CJNE A, #NEG_EDGE, RCV_START_1
                      433     ;
0175 C3               434                    CLR C
0176 7415             435                    MOV A, #HALF_BIT_LOW
0178 25EB             436                    ADD A, CCAP1L
017A F5EB             437                    MOV CCAP1L, A
017C 7401             438                    MOV A, #HALF_BIT_HIGH
017E 35FB             439                    ADDC A, CCAPTH
0180 F5FB             440                    MOV CCAP1H, A
0182 75DB49           441                    MOV CCAPM1, #S_W_TIMER
0185 D0D0             442                    POP PSW
0187 D0E0             443                    POP ACC
0189 32               444                    RETI
                      445     ;
018A B4494B           446     RCV_START_1:   CJNE A, #S_W_TIMER, ERROR_1
018D 20081A           447                    JB RCV_START_BIT_1, RCV_BYTE_1
0190 209445           448                    JB P1.4, ERROR_1
                      449     ;
0193 D208             450                    SETB RCV_START_BIT_1
0195 754209           451                    MOV RCV_COUNT_I, #09H
                      452     ;
0198 C3               453                    CLR C
0199 742C             454                    MOV A, #FULL_BIT_LOW
019B 25EB             455                    ADD A, CCAP1L
019D F5EB             456                    MOV CCAP1L, A
019F 7402             457                    MOV A, #FULL_BIT_HIGH
01A1 35FB             458                    ADDC A, CCAPIH
01A3 F5FB             459                    MOV CCAP1H, A
01A5 D0D0             460                    POP PSW
01A7 D0E0             461                    POP ACC
01A9 32               462                    RETI
                      463     ;
01AA D54212           464     RCV_BYTE_1:    DJNZ RCV_COUNT_1, RCV_DATA_1
                      465     ;
01AD 309428           466     RCV_STOP_1:    JNB P1.4, ERROR_1
01B0 854140           467                    MOV RCV_BUF_1, RCV_REG_1
01B3 D209             468                    SETB RCV_DONE_1
01B5 D28F             469                    SETB TF1
01B7 75DB11           470                    MOV CCAPM1, #NEG_EDGE
01BA D0D0             471                    POP PSW
01BC D0E0             472                    POP ACC
```

```
LOC  OBJ            LINE     SOURCE

01BE 32              473                 RETI
                     474     ;
01BF A294            475     RCV_DATA_1: MOV C, P1.4
01C1 E541            476                 MOV A, RCV_REG_1
01C3 13              477                 RRC A
01C4 F541            478                 MOV RCV_REG_1, A
                     479     ;
01C6 C3              480                 CLR C
01C7 742C            481                 MOV A, #FULL_BIT_LOW
01C9 25EB            482                 ADD A, CCAP1L
01CB F5EB            483                 MOV CCAP1L, A
01CD 7402            484                 MOV A, #FULL_BIT_HIGH
01CF 35FB            485                 ADDC A, CCAP1H
01D1 F5FB            486                 MOV CCAP1H, A
01D3 D0D0            487                 POP PSW
01D5 D0E0            488                 POP ACC
01D7 32              489                 RETI
                     490     ;
01D8 C2B6            491     ERROR_1:    CLR P3.6
01DA 75DB11          492                 MOV CCAPM1, #NEG_EDGE
01DD C208            493                 CLR RCV_START_BIT_1
01DF D0D0            494                 POP PSW
01E1 D0E0            495                 POP ACC
01E3 32              496                 RETI
                     497     ;
                     498     ;
                     499     ;
                     500     ;                          CHANNEL 2
                     501     ;                          ---------
                     502     ;
                     503     ;
01E4 C2DA            504     MODULE_2:   CLR CCF2                    ; Similar to module 0
01E6 E5DC            505                 MOV A, CCAPM2
01E8 547F            506                 ANL A, #01111111B
01EA B41115          507                 CJNE A, #NEG_EDGE, RCV_START_2
                     508     ;
01ED C3              509                 CLR C
01EE 7415            510                 MOV A, #HALF_BIT_LOW
01F0 25EC            511                 ADD A, CCAP2L
01F2 F5EC            512                 MOV CCAP2L, A
01F4 7401            513                 MOV A, #HALF_BIT_HIGH
01F6 35FC            514                 ADDC A, CCAP2H
01F8 F5FC            515                 MOV CCAP2H, A
01FA 75DC49          516                 MOV CCAPM2, #S_W_TIMER
01FD D0D0            517                 POP PSW
01FF D0E0            518                 POP ACC
0201 32              519                 RETI
                     520     ;
0202 B4494B          521     RCV_START_2: CJNE A, #S_W_TIMER, ERROR_2
0205 20101A          522                 JB RCV_START_BIT_2, RCV_BYTE_2
0208 209545          523                 JB P1.5, ERROR_2
                     524     ;
020B D210            525                 SETB RCV_START_BIT_2
020D 755209          526                 MOV RCV_COUNT_2, #09H
                     527     ;
```

270531-17

```
MCS-51 MACRO ASSEMBLER    SWPORT                                    01/01/80   PAGE    8

LOC  OBJ            LINE     SOURCE

0210 C3              528                  CLR C
0211 742C            529                  MOV A, #FULL_BIT_LOW
0213 25EC            530                  ADD A, CCAP2L
0215 F5EC            531                  MOV CCAP2L, A
0217 7402            532                  MOV A, #FULL_BIT_HIGH
0219 35FC            533                  ADDC A, CCAP2H
021B F5FC            534                  MOV CCAP2H, A
021D D0D0            535                  POP PSW
021F D0E0            536                  POP ACC
0221 32              537                  RETI
                     538  ;
0222 D55212          539  RCV_BYTE_2:     DJNZ RCV_COUNT_2, RCV_DATA_2
                     540  ;
0225 309528          541  RCV_STOP_2:     JNB P1.5, ERROR_2
0228 855150          542                  MOV RCV_BUF_2, RCV_REG_2
022B D211            543                  SETB RCV_DONE_2
022D D28F            544                  SETB TF1
022F 75DC11          545                  MOV CCAPM2, #NEG_EDGE
0232 D0D0            546                  POP PSW
0234 D0E0            547                  POP ACC
0236 32              548                  RETI
                     549  ;
0237 A295            550  RCV_DATA_2:     MOV C, P1.5
0239 E551            551                  MOV A, RCV_REG_2
023B 13              552                  RRC A
023C F551            553                  MOV RCV_REG_2, A
023E C3              554                  CLR C
023F 742C            555                  MOV A, #FULL_BIT_LOW
0241 25EC            556                  ADD A, CCAP2L
0243 F5EC            557                  MOV CCAP2L, A
0245 7402            558                  MOV A, #FULL_BIT_HIGH
0247 35FC            559                  ADDC A, CCAP2H
0249 F5FC            560                  MOV CCAP2H, A
024B D0D0            561                  POP PSW
024D D0E0            562                  POP ACC
024F 32              563                  RETI
                     564  ;
0250 C2B7            565  ERROR_2:        CLR P3.7
0252 75DC11          566                  MOV CCAPM2, #NEG_EDGE
0255 C210            567                  CLR RCV_START_BIT_2
0257 D0D0            568                  POP PSW
0259 D0E0            569                  POP ACC
025B 32              570                  RETI
                     571  ;
                     572  ;
                     573  ;
                     574  ;
                     575  ; This routine simulates the "RI" interrupt.  When a byte is received on one
                     576  ; of the channels, this interrupt is generated.  Bits are set so the main
                     577  ; routine knows which channel received a byte.
                     578  ;
                     579  ;
025C C0E0            580  RECEIVE_DONE:   PUSH ACC
025E C0D0            581                  PUSH PSW
0260 C28F            582                  CLR TF1
```

270531-18

```
LOC  OBJ            LINE     SOURCE

0262 300106          583                 JNB RCV_DONE_0, RCV_1   ; Check which module received a byte
0265 C201            584                 CLR RCV_DONE_0          ; Clear flags needed for next reception
0267 C200            585                 CLR RCV_START_BIT_0
0269 D202            586                 SETB RCV_ON_0           ; Tell main routine which channel received
                     587                                         ; a byte
026B 300906          588     RCV_1:      JNB RCV_DONE_1, RCV_2
026E C209            589                 CLR RCV_DONE_1
0270 C208            590                 CLR RCV_START_BIT_1
0272 D20A            591                 SETB RCV_ON_1
                     592     ;
0274 301106          593     RCV_2:      JNB RCV_DONE_2, RETURN
0277 C211            594                 CLR RCV_DONE_2
0279 C210            595                 CLR RCV_START_BIT_2
027B D212            596                 SETB RCV_ON_2
                     597     ;
027D D0D0            598     RETURN:     POP PSW
027F D0E0            599                 POP ACC
0281 32              600                 RETI
                     601     ;
                     602     ;
                     603     ;
                     604     ;
                     605     ;           SERIAL PORT INTERRUPT
                     606     ;           ====================
                     607     ;
                     608     ; When a byte is received on the full-duplex serial port, it is then
                     609     ; transmitted back to a "dummy" terminal.  This terminal checks that the
                     610     ; byte it transmitted to the PCA is the same value it receives back.
                     611     ;
                     612     ;
0282 C0E0            613     SERIAL_PORT:    PUSH ACC
0284 C0D0            614                     PUSH PSW
0286 30980B          615                     JNB RI, TXM             ; Check whether RI or TI
0289 E599            616                     MOV A, SBUF             ; caused the interrupt
028B C298            617                     CLR RI
028D F599            618                     MOV SBUF, A
028F D0D0            619                     POP PSW
0291 D0E0            620                     POP ACC
0293 32              621                     RETI
                     622     ;
0294 C299            623     TXM:            CLR TI
0296 D0D0            624                     POP PSW
0298 D0E0            625                     POP ACC
029A 32              626                     RETI
                     627     ;
                     628     END

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

270531-19

```
MCS-51 MACRO ASSEMBLER    SWPORT                                                01/01/80   PAGE    1


DOS 3.20 (038-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN SWPORT.OBJ
ASSEMBLER INVOKED BY:  C:\AEDIT\ASM51.EXE SWPORT.TR


LOC  OBJ            LINE     SOURCE

                       1     $NOMOD51
                       2     $NOSYMBOLS
                       3     $NOLIST
                     152     ;
                     153     ;
                     154     ; This program tests the transmit routines for the software serial port.
                     155     ; To initialize the first transmission, the compare values are loaded before
                     156     ; the PCA timer is started.  Successive interrupts are generated every bit
                     157     ; time by the software timer.
                     158     ;
                     159     ; For test purposes, the data transmitted increments from 00 to FF hex.
                     160     ; "Dummy" terminals receive these bytes and display the bytes as they
                     161     ; are incremented.
                     162     ;
                     163     ;
0000                 164     ORG 00H
0000 020036          165     LJMP INIT_TXM
                     166     ;
0023                 167     ORG 0023H
0023 02014B          168     LJMP SERIAL_PORT                        ; Serial port interrupt
                     169     ;
0033                 170     ORG 0033H
0033 0200D0          171     LJMP TRANSMIT                           ; PCA software timer interrupt
                     172     ;
                     173     ;
                     174     ;                       VARIABLES USED BY THE SOFTWARE SERIAL PORT
                     175     ;                       ==========================================
                     176     ;
  0003               177     TXM_START_BIT_0         BIT             20H.3   ; Indicates start bit has been
  000B               178     TXM_START_BIT_1         BIT             21H.3   ; transmitted
  0013               179     TXM_START_BIT_2         BIT             22H.3
                     180     ;
  0004               181     TXM_IN_PROGRESS_0       BIT             20H.4   ; Indicates transmit is in progress
  000C               182     TXM_IN_PROGRESS_1       BIT             21H.4
  0014               183     TXM_IN_PROGRESS_2       BIT             22H.4
                     184     ;
  0034               185     TXM_BUF_0               DATA            34H     ; Software transmit "SBUF"
  0044               186     TXM_BUF_1               DATA            44H
  0054               187     TXM_BUF_2               DATA            54H
                     188     ;
  0035               189     TXM_REG_0               DATA            35H     ; Temporary register for
  0045               190     TXM_REG_1               DATA            45H     ; transmitting bits
  0055               191     TXM_REG_2               DATA            55H
                     192     ;
  0036               193     TXM_COUNT_0             DATA            36H     ; Counter for transmitting bits
  0046               194     TXM_COUNT_1             DATA            46H
  0056               195     TXM_COUNT_2             DATA            56H
                     196     ;
  0037               197     DATA_0                  DATA            37H     ; Register used for the test
  0047               198     DATA_1                  DATA            47H     ; program
```

270531-20

```
LOC  OBJ            LINE     SOURCE

  0057              199      DATA_2                DATA            .57H
                    200      ;
  0049              201      S_W_TIMER             EQU             49H     ; Software timer mode for the
                    202                                                    ; compare/capture module
  002C              203      FULL_BIT_LOW          EQU             2CH     ; Full bit time = 22CH
  0002              204      FULL_BIT_HIGH         EQU             02H     ; 2400 Baud at 16 MHz
                    205      ;
                    206      ;
                    207      ;                     INITIALIZATION
                    208      ;                     ==============
                    209      ;
0036 75815F         210      INIT_TXM:       MOV SP, #5FH                  ; (Compatible with receive routines)
                    211      ;
0039 75D900         212                      MOV CMOD, #00H                ; Increment PCA timer @ 1/12 osc. freq.
003C 75D800         213                      MOV CCON, #00H                ; Clear all status flags
003F 75F900         214                      MOV CH, #00H
0042 75E900         215                      MOV CL, #00H
0045 75DD49         216                      MOV CCAPM3, #S_W_TIMER        ; Module 3 configured as software timer
                    217      ;
0048 75A8D8         218                      MOV IE, #0D8H                 ; Initialize all needed interrupts
                    219
004B 759850         220      INIT_SP:        MOV SCON, #50H                ; Serial port in mode 1 (8-bit UART)
004E 75CBFF         221                      MOV RCAP2H, #0FFH             ; Reload values for 9600 Baud @ 16 MHz
0051 75CACC         222                      MOV RCAP2L, #0CCH
0054 75C834         223                      MOV T2CON, #34H               ; Timer 2 as a baud-rate generator,
                    224                                                    ; turn Timer 2 on
                    225      ;
0057 C203           226      INIT_FLAGS:     CLR TXM_START_BIT_0
0059 C20B           227                      CLR TXM_START_BIT_1
005B C213           228                      CLR TXM_START_BIT_2
                    229      ;
005D C204           230                      CLR TXM_IN_PROGRESS_0
005F C20C           231                      CLR TXM_IN_PROGRESS_1
0061 C214           232                      CLR TXM_IN_PROGRESS_2
                    233      ;
0063 753400         234                      MOV TXM_BUF_0, #00H
0066 754400         235                      MOV TXM_BUF_1, #00H
0069 755400         236                      MOV TXM_BUF_2, #00H
                    237      ;
006C 753500         238                      MOV TXM_REG_0, #00H
006F 754500         239                      MOV TXM_REG_1, #00H
0072 755500         240                      MOV TXM_REG_2, #00H
                    241      ;
0075 753600         242                      MOV TXM_COUNT_0, #00H
0078 754600         243                      MOV TXM_COUNT_1, #00H
007B 755600         244                      MOV TXM_COUNT_2, #00H
                    245      ;
007E 7537FF         246                      MOV DATA_0, #0FFH
0081 7547FF         247                      MOV DATA_1, #0FFH
0084 7557FF         248                      MOV DATA_2, #0FFH
                    249      ;
0087 75ED2C         250                      MOV CCAP3L, #2CH              ; Cause the first software timer to
008A 75FD02         251                      MOV CCAP3H, #02H              ; interrupt one bit time after
008D D2DE           252                      SETB CR                       ; PCA timer is started
                    253      ;
```

270531-21

```
MCS-51 MACRO ASSEMBLER    SWPORT                                    01/01/80   PAGE    3

LOC  OBJ            LINE     SOURCE

                     254     ;
                     255     ;                       MAIN TEST ROUTINE - TRANSMIT BITS
                     256     ;                       =================================
                     257     ;
008F 02009D          258     FIRST_TXM:      JMP TXM_ON_0
                     259     ;
0092 300408          260     MAIN_TXM:       JNB TXM_IN_PROGRESS_0, TXM_ON_0 ; Determine if ready to send
0095 300C16          261                     JNB TXM_IN_PROGRESS_1, TXM_ON_1 ; next byte. (i.e. transmit
0098 301424          262                     JNB TXM_IN_PROGRESS_2, TXM_ON_2 ; "not" in progress)
009B 80F5            263                     JMP MAIN_TXM                    ; Waiting for "TI" flag
                     264     ;
                     265     ;
009D C203            266     TXM_ON_0:       CLR TXM_START_BIT_0             ; Clear flag from previous
009F 0537            267                     INC DATA_0                      ; transmission
00A1 853734          268                     MOV TXM_BUF_0, DATA_0           ; Load "SBUF" with data byte
00A4 853435          269                     MOV TXM_REG_0, TXM_BUF_0
00A7 753609          270                     MOV TXM_COUNT_0, #09H           ; 8 data bits + 1 stop bit
00AA D204            271                     SETB TXM_IN_PROGRESS_0
00AC 80E4            272                     JMP MAIN_TXM
                     273     ;
00AE C20B            274     TXM_ON_1:       CLR TXM_START_BIT_1
00B0 0547            275                     INC DATA_1
00B2 854744          276                     MOV TXM_BUF_1, DATA_1
00B5 854445          277                     MOV TXM_REG_1, TXM_BUF_1
00B8 754609          278                     MOV TXM_COUNT_1, #09H
00BB D20C            279                     SETB TXM_IN_PROGRESS_1
00BD 80D3            280                     JMP MAIN_TXM
                     281     ;
00BF C213            282     TXM_ON_2:       CLR TXM_START_BIT_2
00C1 0557            283                     INC DATA_2
00C3 855754          284                     MOV TXM_BUF_2, DATA_2
00C6 855455          285                     MOV TXM_REG_2, TXM_BUF_2
00C9 755609          286                     MOV TXM_COUNT_2, #09H
00CC D214            287                     SETB TXM_IN_PROGRESS_2
00CE 80C2            288                     JMP MAIN_TXM
                     289     ;
                     290     ;
                     291     ;                       PCA INTERRUPT ROUTINE - TRANSMIT BITS
                     292     ;                       =====================================
                     293     ;
00D0 C0E0            294     TRANSMIT:       PUSH ACC
00D2 C0D0            295                     PUSH PSW
00D4 C2DB            296                     CLR CCF3                        ; Clear s/w timer interrupt
00D6 30041E          297                     JNB TXM_IN_PROGRESS_0,TRANSMIT_1  ; Check which channel is
                     298                                                     ; transmitting
                     299     ;
                     300     ;                               CHANNEL 0
                     301     ;                               ---------
                     302
00D9 200307          303     TRANSMIT_0:     JB TXM_START_BIT_0, TXM_BYTE_0  ; If start bit has been sent,
                     304                                                     ; continue transmitting data bits,
00DC C2B2            305                     CLR P3.2                        ; otherwise transmit start bit
00DE D203            306                     SETB TXM_START_BIT_0            ; Signify start bit sent
00E0 0200F7          307                     JMP TRANSMIT_1                  ; Check next transmit pin
                     308     ;
```

270531-22

```
LOC  OBJ            LINE     SOURCE

00E3 D53607          309     TXM_BYTE_0:     DJNZ TXM_COUNT_0, TXM_DATA_0    ; If bit count equals 1 thru 9,
                     310                                                     ; Transmit data bits (8 total)
00E6 D2B2            311     TXM_STOP_0:     SETB P3.2                       ; When bit count = 0, transmit stop bit
00E8 C204            312                     CLR TXM_IN_PROGRESS_0           ; Indicate transmission is finished and
                     313                                                     ; ready for next byte
00EA 0200F7          314                     JMP TRANSMIT_1                  ; Check next transmit pin
                     315     ;
00ED E535            316     TXM_DATA_0:     MOV A, TXM_REG_0                ; Transmit one bit at a time
00EF 13              317                     RRC A                           ; through the carry bit
00F0 92B2            318                     MOV P3.2, C
00F2 F535            319                     MOV TXM_REG_0, A                ; Save what's not been sent
00F4 0200F7          320                     JMP TRANSMIT_1                  ; Check next transmit pin
                     321     ;
                     322     ;                               CHANNEL 1
                     323     ;                               ---------
                     324     ;
00F7 300C1E          325     TRANSMIT_1:     JNB TXM_IN_PROGRESS_1, TRANSMIT_2 ; Similar to TRANSMIT_0
00FA 200B07          326                     JB TXM_START_BIT_1, TXM_BYTE_1
00FD C2B3            327                     CLR P3.3
00FF D20B            328                     SETB TXM_START_BIT_1
0101 020118          329                     JMP TRANSMIT_2
                     330     ;
0104 D54607          331     TXM_BYTE_1:     DJNZ TXM_COUNT_1, TXM_DATA_1
                     332     ;
0107 D2B3            333     TXM_STOP_1:     SETB P3.3
0109 C20C            334                     CLR TXM_IN_PROGRESS_1
010B 020118          335                     JMP TRANSMIT_2
                     336     ;
010E E545            337     TXM_DATA_1:     MOV A, TXM_REG_1
0110 13              338                     RRC A
0111 92B3            339                     MOV P3.3, C
0113 F545            340                     MOV TXM_REG_1, A
0115 020118          341                     JMP TRANSMIT_2
                     342     ;
                     343     ;                               CHANNEL 2
                     344     ;                               ---------
                     345     ;
0118 30141E          346     TRANSMIT_2:     JNB TXM_IN_PROGRESS_2, TXM_EXIT ; Similar to TRANSMIT_0
011B 201307          347                     JB TXM_START_BIT_2, TXM_BYTE_2
011E C2B4            348                     CLR P3.4
0120 D213            349                     SETB TXM_START_BIT_2
0122 020139          350                     JMP TXM_EXIT
                     351     ;
0125 D55607          352     TXM_BYTE_2:     DJNZ TXM_COUNT_2, TXM_DATA_2
                     353     ;
0128 D2B4            354     TXM_STOP_2:     SETB P3.4
012A C214            355                     CLR TXM_IN_PROGRESS_2
012C 020139          356                     JMP TXM_EXIT
                     357     ;
012F E555            358     TXM_DATA_2:     MOV A, TXM_REG_2
0131 13              359                     RRC A
0132 92B4            360                     MOV P3.4, C
0134 F555            361                     MOV TXM_REG_2, A
0136 020139          362                     JMP TXM_EXIT
                     363     ;
```

270531-23

```
MCS-51 MACRO ASSEMBLER    SWPORT                                                01/01/80   PAGE    5

LOC  OBJ            LINE     SOURCE

0139 C3              364     TXM_EXIT:       CLR C                           ; Update compare value with
013A 742C            365                     MOV A, #FULL_BIT_LOW            ; full bit time = 22CH
013C 25ED            366                     ADD A, CCAP3L
013E F5ED            367                     MOV CCAP3L, A
0140 7402            368                     MOV A, #FULL_BIT_HIGH
0142 35FD            369                     ADDC A, CCAP3H
0144 F5FD            370                     MOV CCAP3H, A
0146 D0D0            371                     POP PSW
0148 D0E0            372                     POP ACC
014A 32              373                     RETI
                     374     ;
                     375     ;
                     376     ;               SERIAL PORT INTERRUPT
                     377     ;               ====================
                     378     ;
                     379     ; When a byte is received on the full-duplex serial port, it is then
                     380     ; transmitted back to a "dummy" terminal.  This terminal checks that
                     381     ; the byte it transmitted to the PCA is the same value it receives back.
                     382     ;
                     383     ;
014B C0E0            384     SERIAL_PORT:    PUSH ACC
014D C0D0            385                     PUSH PSW
014F 30980B          386                     JNB RI, TXM                     ; Check whether RI or TI
0152 E599            387                     MOV A, SBUF                     ; caused the interrupt
0154 C298            388                     CLR RI
0156 F599            389                     MOV SBUF, A
0158 D0D0            390                     POP PSW
015A D0E0            391                     POP ACC
015C 32              392                     RETI
                     393     ;
015D C299            394     TXM:            CLR TI
015F D0D0            395                     POP PSW
0161 D0E0            396                     POP ACC
0163 32              397                     RETI
                     398     ;
                     399     ;
                     400     END

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

270531-24