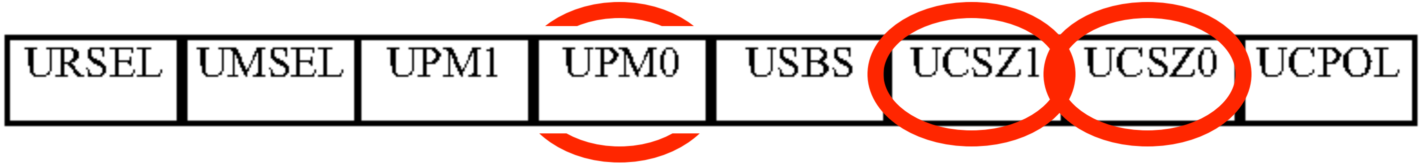
Serial Port Programming



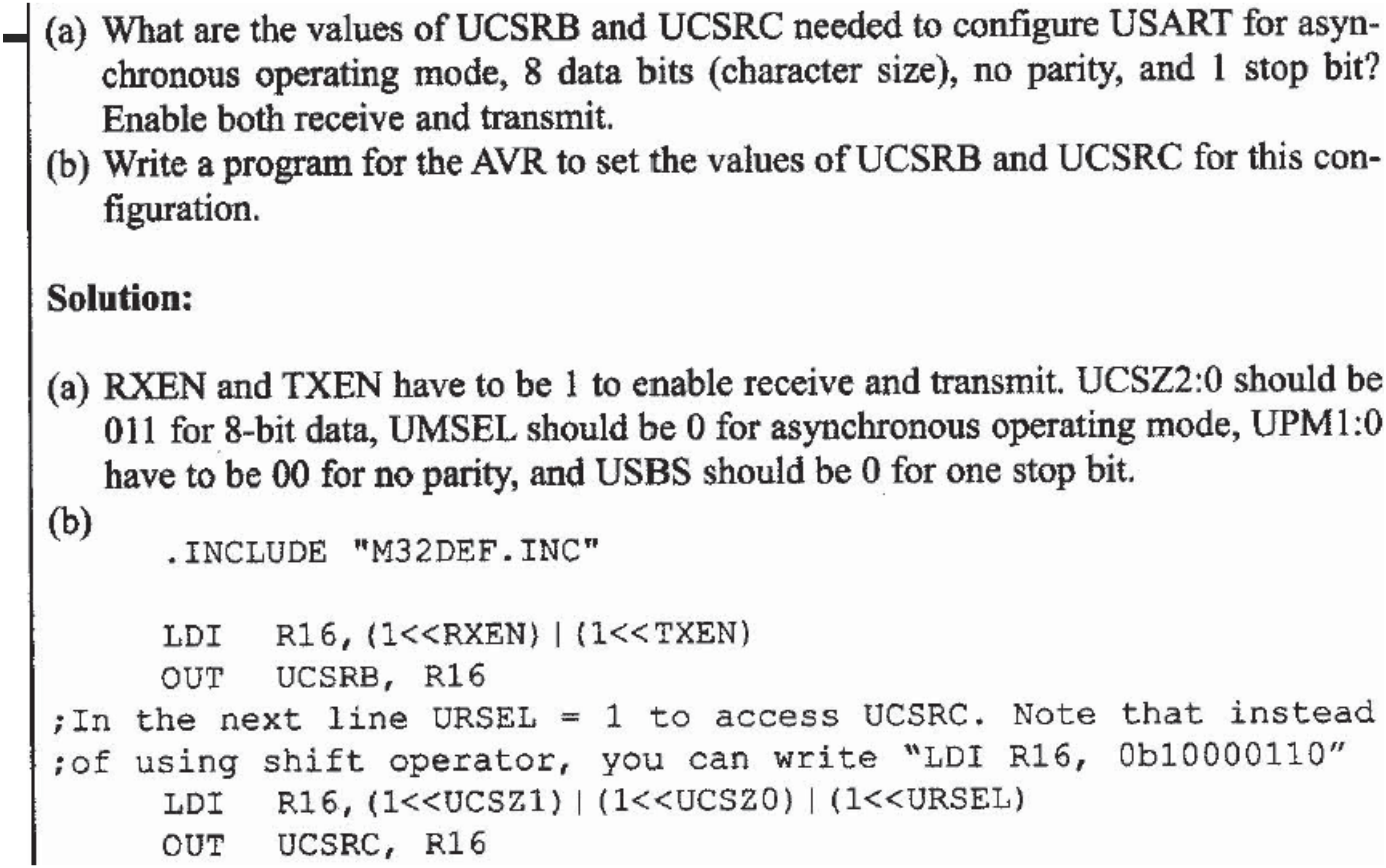
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

1. The UCSRB register is loaded with the value 10H, enabaling USART receiver. The receiver will override normal port operation for the RxD pin when enabled.
2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity and one stop bit.
3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 8 MHz) to set the baud rate for serial data transfer.
5. The RXC flag bit of the UCSRA register is monitored for a HIGH to see if an entire character has been received yet.
6. When RXC is raised, the UDR register has the byte. Its contents are moved into a safe place.
7. To receive the next character, go to Step 5.

Serial Port Programming

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

1. The UCSRB register is loaded with the value 08H, enabaling USART transmitter. The Transmitter will override normal port operation for the TxD pin when enabled.
2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity and one stop bit.
3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 4 MHz) to set the baud rate for serial data transfer.
4. The character byte to be transmitted serially is written into the UDR register.
6. Monitor the UDRE bit of the UCSRA register to make sure UDR is ready for next byte.
7. To transfer the next character, go to Step 5.



(a) What are the values of UCSRB and UCSRC needed to configure USART for asynchronous operating mode, 8 data bits (character size), no parity, and 1 stop bit? Enable both receive and transmit.

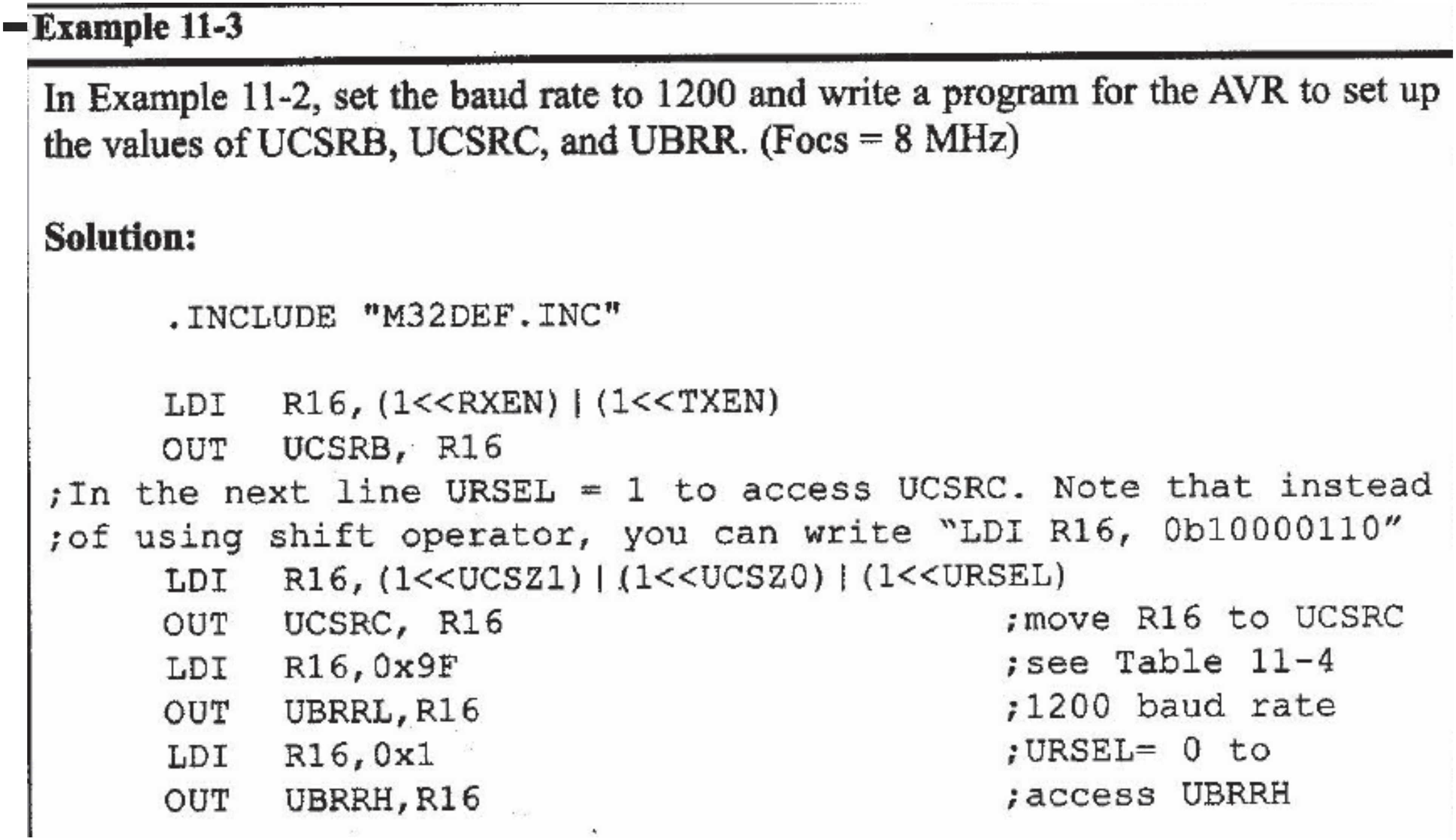
(b) Write a program for the AVR to set the values of UCSRB and UCSRC for this configuration.

Solution:

(a) RXEN and TXEN have to be 1 to enable receive and transmit. UCSZ2:0 should be 011 for 8-bit data, UMSEL should be 0 for asynchronous operating mode, UPM1:0 have to be 00 for no parity, and USBS should be 0 for one stop bit.

(b)

```
      .INCLUDE "M32DEF.INC"

      LDI   R16,(1<<RXEN)|(1<<TXEN)
      OUT   UCSRB, R16
;In the next line URSEL = 1 to access UCSRC. Note that instead
;of using shift operator, you can write "LDI R16, 0b10000110"
      LDI   R16,(1<<UCSZ1)|(1<<UCSZ0)|(1<<URSEL)
      OUT   UCSRC, R16
```



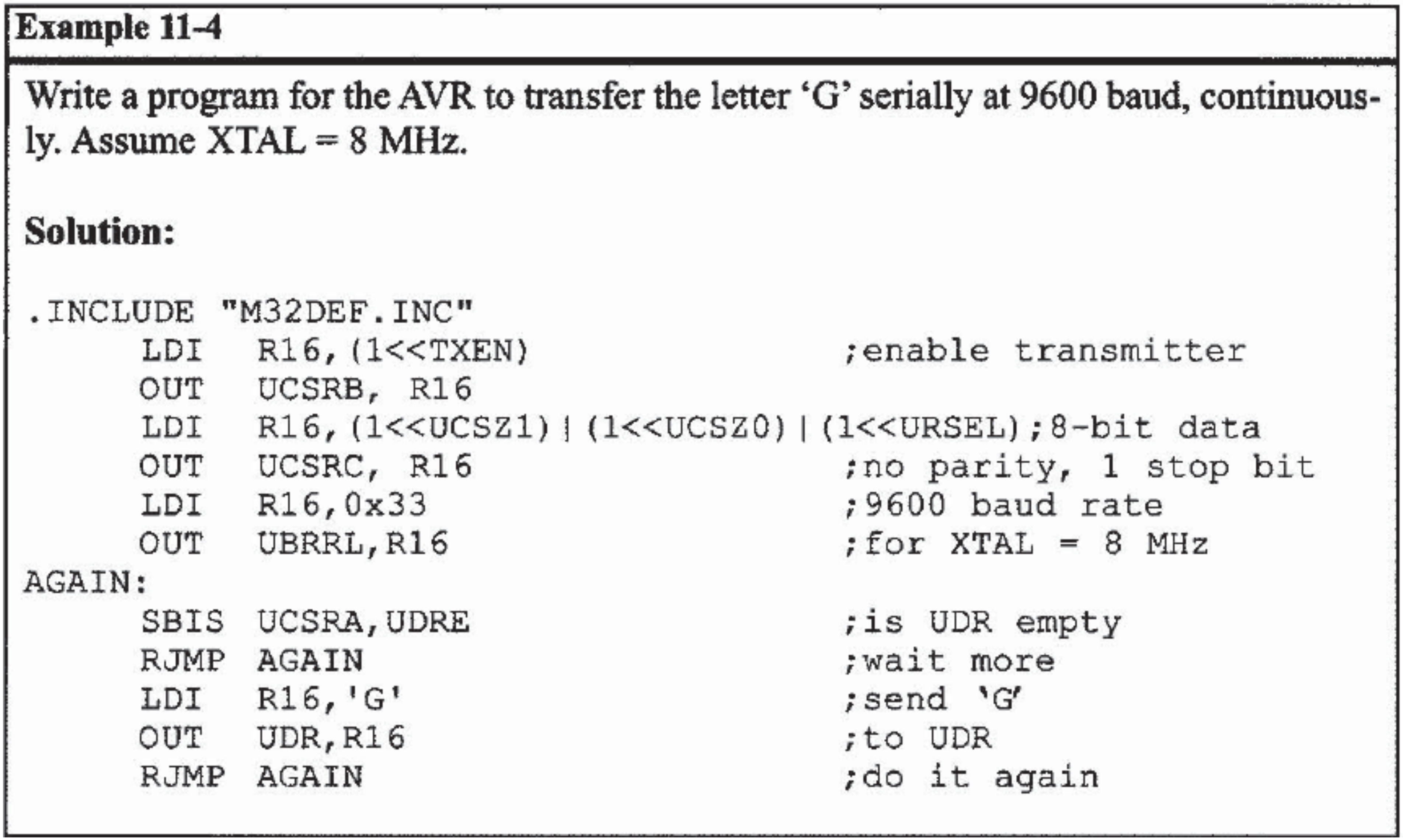
Example 11-3

In Example 11-2, set the baud rate to 1200 and write a program for the AVR to set up the values of UCSRB, UCSRC, and UBRR. (Focs = 8 MHz)

Solution:

```
      .INCLUDE "M32DEF.INC"

      LDI   R16,(1<<RXEN)|(1<<TXEN)
      OUT   UCSRB, R16
;In the next line URSEL = 1 to access UCSRC. Note that instead
;of using shift operator, you can write "LDI R16, 0b10000110"
      LDI   R16,(1<<UCSZ1)|(1<<UCSZ0)|(1<<URSEL)
      OUT   UCSRC, R16                      ;move R16 to UCSRC
      LDI   R16,0x9F                        ;see Table 11-4
      OUT   UBRRL,R16                       ;1200 baud rate
      LDI   R16,0x1                         ;URSEL= 0 to
      OUT   UBRRH,R16                       ;access UBRRH
```

Serial Tranmit



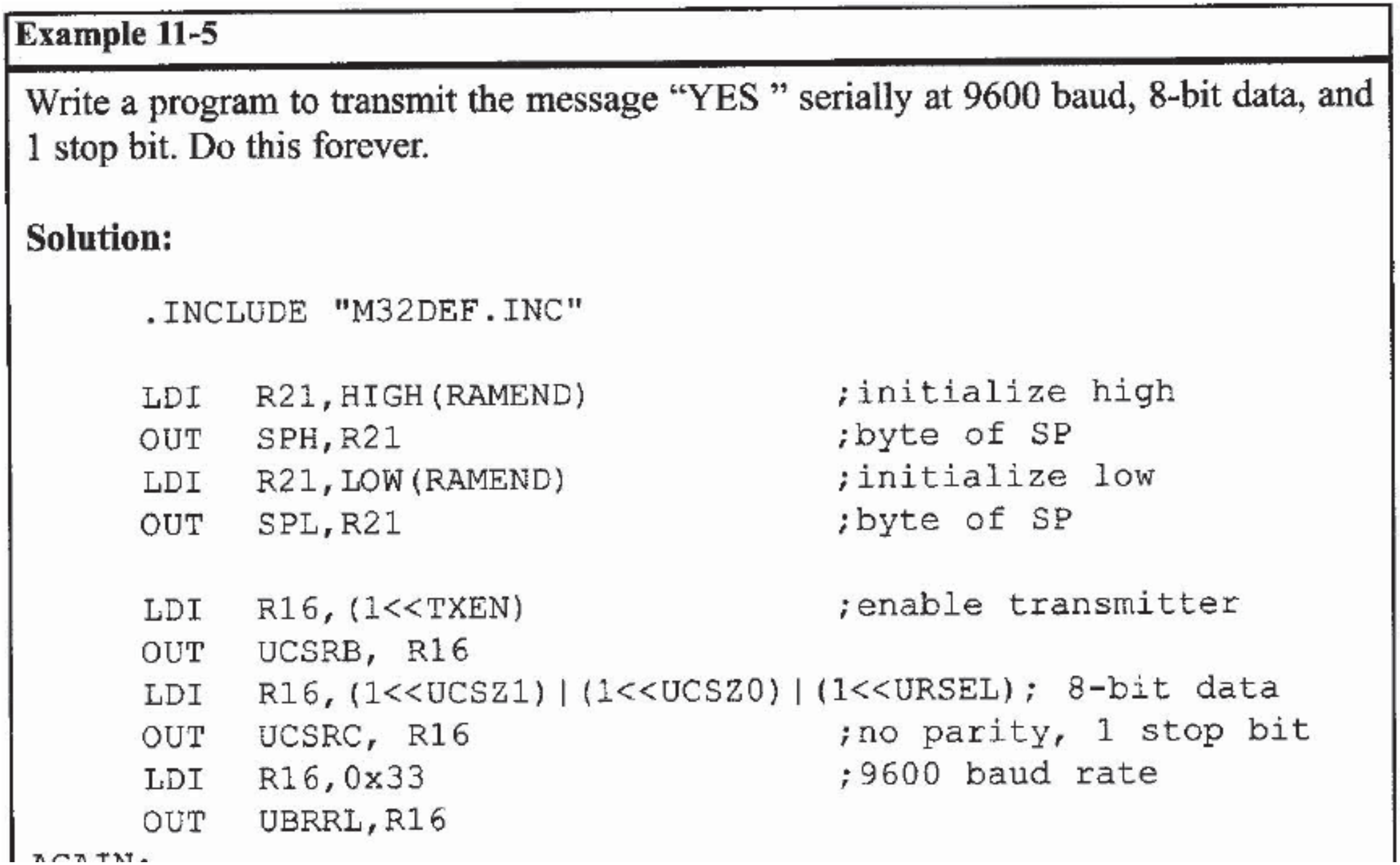
Example 11-4

Write a program for the AVR to transfer the letter ‘G’ serially at 9600 baud, continuously. Assume XTAL = 8 MHz.

Solution:

```
.INCLUDE "M32DEF.INC"
      LDI   R16,(1<<TXEN)                       ;enable transmitter
      OUT   UCSRB, R16
      LDI   R16,(1<<UCSZ1)|(1<<UCSZ0)|(1<<URSEL);8-bit data
      OUT   UCSRC, R16                          ;no parity, 1 stop bit
      LDI   R16,0x33                            ;9600 baud rate
      OUT   UBRRL,R16                           ;for XTAL = 8 MHz
AGAIN:
      SBIS  UCSRA,UDRE                          ;is UDR empty
      RJMP  AGAIN                               ;wait more
      LDI   R16,'G'                             ;send ‘G’
      OUT   UDR,R16                             ;to UDR
      RJMP  AGAIN                               ;do it again
```

Transmit "YES"

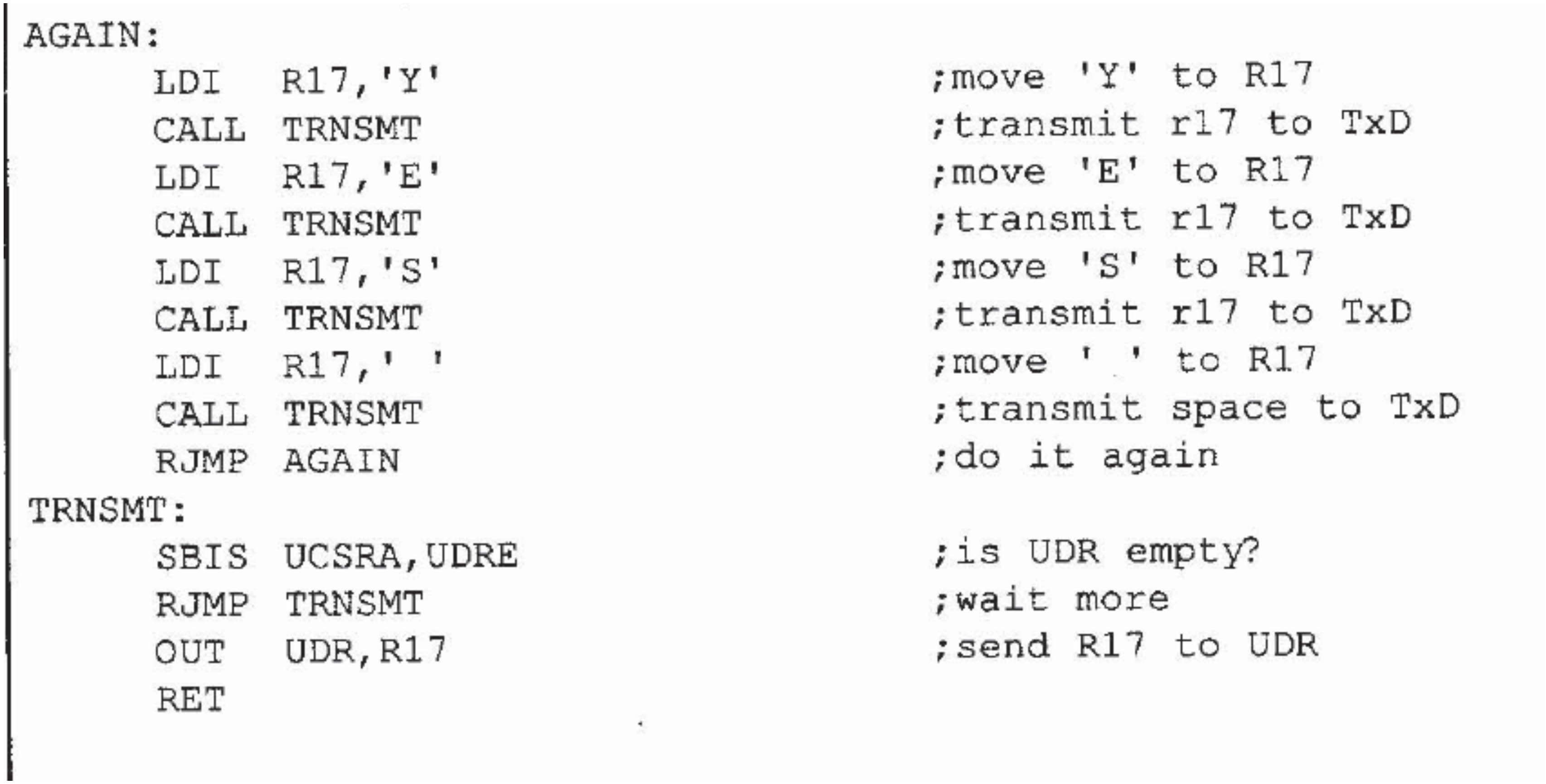


Example 11-5

Write a program to transmit the message "YES " serially at 9600 baud, 8-bit data, and 1 stop bit. Do this forever.

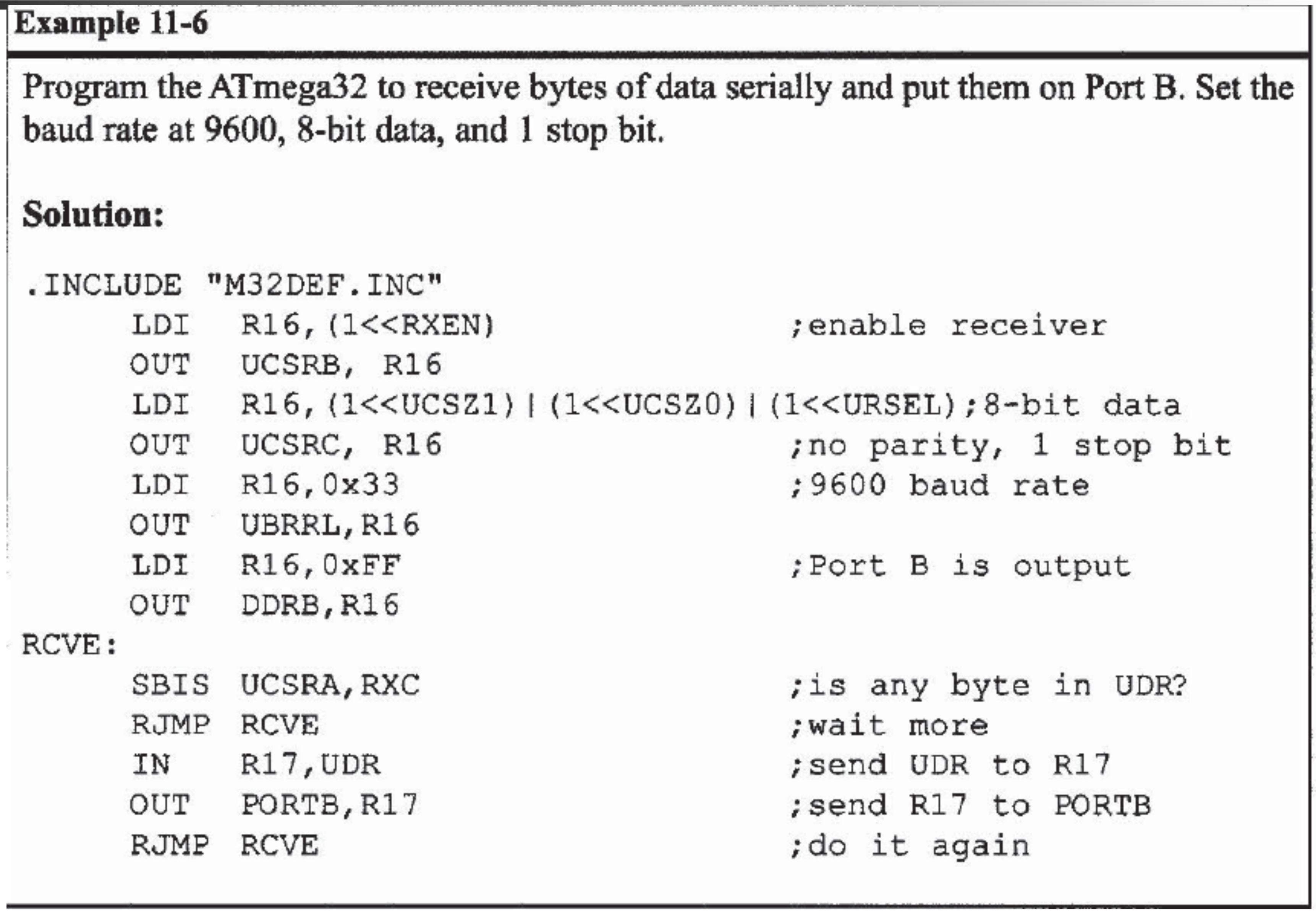
Solution:

```
       .INCLUDE "M32DEF.INC"

       LDI    R21,HIGH(RAMEND)              ;initialize high
       OUT    SPH,R21                       ;byte of SP
       LDI    R21,LOW(RAMEND)               ;initialize low
       OUT    SPL,R21                       ;byte of SP

       LDI    R16,(1<<TXEN)                 ;enable transmitter
       OUT    UCSRB, R16
       LDI    R16,(1<<UCSZ1)|(1<<UCSZ0)|(1<<URSEL); 8-bit data
       OUT    UCSRC, R16                    ;no parity, 1 stop bit
       LDI    R16,0x33                      ;9600 baud rate
       OUT    UBRRL,R16
AGAIN:
```



```
AGAIN:
      LDI   R17,'Y'                        ;move 'Y' to R17
      CALL  TRNSMT                         ;transmit r17 to TxD
      LDI   R17,'E'                        ;move 'E' to R17
      CALL  TRNSMT                         ;transmit r17 to TxD
      LDI   R17,'S'                        ;move 'S' to R17
      CALL  TRNSMT                         ;transmit r17 to TxD
      LDI   R17,' '                        ;move ' ' to R17
      CALL  TRNSMT                         ;transmit space to TxD
      RJMP  AGAIN                          ;do it again
TRNSMT:
      SBIS  UCSRA,UDRE                     ;is UDR empty?
      RJMP  TRNSMT                         ;wait more
      OUT   UDR,R17                        ;send R17 to UDR
      RET
```

Receive Serially



Example 11-6

Program the ATmega32 to receive bytes of data serially and put them on Port B. Set the baud rate at 9600, 8-bit data, and 1 stop bit.

Solution:

```
.INCLUDE "M32DEF.INC"
      LDI   R16,(1<<RXEN)                   ;enable receiver
      OUT   UCSRB, R16
      LDI   R16,(1<<UCSZ1)|(1<<UCSZ0)|(1<<URSEL);8-bit data
      OUT   UCSRC, R16                      ;no parity, 1 stop bit
      LDI   R16,0x33                        ;9600 baud rate
      OUT   UBRRL,R16
      LDI   R16,0xFF                        ;Port B is output
      OUT   DDRB,R16
RCVE:
      SBIS  UCSRA,RXC                       ;is any byte in UDR?
      RJMP  RCVE                            ;wait more
      IN    R17,UDR                         ;send UDR to R17
      OUT   PORTB,R17                       ;send R17 to PORTB
      RJMP  RCVE                            ;do it again
```