AN547

Serial Port Utilities

Author: *Amar Palacherla*
Microchip Technology Inc.

INTRODUCTION

The PIC17C42 has an on-chip high speed Universal Synchronous Asynchronous Receiver Transmitter (USART). The serial port can be configured to operate either in full-duplex asynchronous mode or half duplex synchronous mode. The serial port has a dedicated 8-bit baud rate generator. Either 8- or 9-bits can be transmitted/received.

This application note provides information on using the serial port, parity generation, serial port expansion, RS-232 interface, I/O port expansion using the serial port in synchronous mode.

SERIAL PORT USAGE

A brief code to setup the serial port, and receive and transmit data is given in Example 1. Small sections of code for both asynchronous and synchronous mode are given.

Asynchronous Mode Setup

Asynchronous mode setup requires selection of 8/9-bits of data transfer, baud rate, setting the baud rate generator, and configuring the TXSTA and RCSTA control registers. The baud rate generator is configured by writing the appropriate value to SPBRG register (bank0, file 17h). The value to be written to SPBRG is given by:

$$SPBRG = \frac{Input\_Clk\_Freq}{64 \bullet Baud\_Rate} - 1$$

For example, to select a baud rate of 9600 bits/sec with input clock frequency of 16 MHz, SPBRG is computed from the above equation to be 25. Once the Baud Rate Generator is set up, it is necessary to configure the TXSTA and RCSTA control registers as shown in Example 1 (refer to the data sheet):

EXAMPLE 1: INITIALIZATION EXAMPLE

```
SPBRG   :   25                  :   9600 baud @ 16 MHz input clock
TXSTA   :   00100000   (20h)   :   8-bit transmission, async mode
RCSTA   :   10010000   (90h)   :   8-bit reception, enable serial port, enable reception

;**********************************************************************
;       Sample Code For Asynchronous Mode Serial Port Setup
;**********************************************************************

#define ClkFreq        16000000                     ; input clock frequency = 16 MHz
#define baud(X)        ((10*ClkFreq/(64*X))+5)/10 - 1
#define TXSTA_INIT     0x90
#define RCSTA_INIT     0x90

    #include           "17C42.h"                    ; file containing the Register Definitions

    Setup_Async_Mode
                movlb   0                           ; SPBRG, TXSTA & RCSTA are in bank 0
                movlw   baud(9600)                  ; equals 25 for 9600 baud
                movwf   SPBRG                       ; baud rate generator is reset & initialized
                movlw   TXSTA_INIT
                movwf   TXSTA                       ; 8-bit transmission, async mode
                movlw   RCSTA_INIT
                movwf   RCSTA                       ; 8-bit reception, enable serial port,
                                                    ; enable reception
                return
;**********************************************************************
```

Synchronous Mode Setup

Synchronous mode setup requires selection of 8/9-bits of data transfer, bit rate, setting the baud rate generator, and configuring the TXSTA and RCSTA control registers. The baud rate generator is configured by writing the appropriate value to SPBRG register (bank0, file 17h). The value to be written to SPBRG is given by:

$$SPBRG = \frac{Input\_Clk\_Freq}{4 \bullet Baud\_Rate} \quad -1$$

For example, to select a bit rate of 1 Mbits/sec with input clock frequency of 16 MHz, SPBRG is computed from the above equation to be 3. Once the Baud Rate Generator is set up, it is necessary to configure the TXSTA and RCSTA control registers as follows (please refer to the data sheet) :

EXAMPLE 2: CONFIGURING TXSTA AND RCSTA CONTROL REGISTORS

```
SPBRG   :   3                   :   1 Mbits/sec @ 16 Mhz input clock
TXSTA   :   10110000   (B0h)   :   8-bit transmission, Sync mode (MASTER)
RCSTA   :   10010000   (90h)   :   8-bit reception, enable serial port, continuous
                                   reception

;**********************************************************************
;          Sample Code For Synchronous Mode (MASTER) Serial Port Setup
;**********************************************************************

#define ClkFreq       16000000                  ; input clock frequency = 16 Mhz
#define baud(X)       ((10*ClkFreq/(4*X))+5)/10 - 1
#define TXSTA_INIT    0xB0
#define RCTSA_INIT    0x90

       #include              "17C42.h"         ; file containing the Register Definitions

Setup_Sync_Master_Mode
               movlb   0                       ; SPBRG, TXSTA & RCSTA are in bank 0
               movlw   baud(1000000)           ; equals 3 for 1 Mbits/sec
               movwf   SPBRG                   ; baud rate generator is reset & initialized
               movlw   TXSTA_INIT
               movwf   TXSTA                   ; 8-bit transmission, async mode
               movlw   RCSTA_INIT
               movwf   RCSTA                   ; 8-bit reception, enable serial port,
                                               ; enable reception
               return
;**********************************************************************
```

Receiving Data (Software Polling)

The sample code in Example 3 provides a way to read the received serial data by software polling (with no serial port interrupts). This applies to both asynchronous and synchronous mode. Software polling is done by checking the RCIF bit (PIR<0>). If this bit is set it means that a word has been received (8 bits are in RCREG and the 9th bit in RCSTA<0>).

EXAMPLE 3: POLLING FOR RECEIVE DATA

```
;*****************************************************************
;          Return The 8-bit received Data By Software Polling
;      The received data is returned in location SerInData
;*****************************************************************
Get_Serial_Data_Poll
               movlb   1                ; PIR is in bank 1
PollRcv        btfss   PIR,0            ; check the RBIF bit
               goto    PollRcv          ; loop until char received, assume WDT is off
               movlb   0                ; RCREG is in bank 0
               movpf   RCREG,SerInData
               return                   ; Received 8-bits are in SerInData
;*****************************************************************
```

Transmitting Data (Software Polling)

The sample code in Example 4 provides a way to transmit serial data by software polling (no serial port interrupts). Software polling is done by checking the bit (PIR<1> in bank 1) to be one, indicating the transfer of TXIF to the serial shift register.

EXAMPLE 4: POLLING FOR TRANSMIT DATA

```
;*****************************************************************
;          Transmit 8-bit  Data By Software Polling
;      The data to be transmitted is in location SerOutData
;*****************************************************************
Send_Serial_Data_Poll
               movlb   1                ; PIR is in bank 1
PollTXIFbtfss  PIR,1                    ; check the TXIF bit of PIR register in bank1
               goto    PollTXIF         ; loop until char received, assume WDT is off
               movlb   0                ; RCREG is in bank 0
               movfp   SerOutData,TXREG
               return                   ; Received 8-bits are in SerInData
;*****************************************************************
```

Transmitting & Receiving A Block Of Data (Interrupt Driven)

A general purpose routine which is interrupt driven, that transmits and receives a block of data is provided in Example 5. The reception or transmission of the block is ended when an end of block character is detected. As an example, the end of block is identified by a 0. The block of data to be transmitted is stored in the program memory and the TABLRD instruction is used to transfer this example data to the file registers and serial port. The user may modify this code to a more general purpose routine that suits the application.

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

EXAMPLE 5: INTERRUPT DRIVEN TRANSMIT/RECEIVE

```
MPASM 01.40 Released           SERINT.ASM   1-22-1997  10:20:49         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001 ;        TITLE   'Serial Interface Routines
                     00002 ;        PROCESSOR     42
                     00003
                     00004 ;This is a short program to demonstrate how to transmit and receive
                     00005 ;serial data using the PIC17C42.
                     00006 ;
                     00007 ;A message will be transmitted and routed right back to the processor
                     00008 ;and read. The read information will be saved in an internal buffer.
                     00009 ;
                     00010 ;       Program:          SERINT.ASM
                     00011 ;       Revision Date:
                     00012 ;                       1-22-97      Compatibility with MPASMWIN 1.40
                     00013 ;
                     00014 ;
                     00015         LIST     P = 17C42
                     00016
                     00017 #include    <p17c42.inc>
                     00001         LIST
                     00002 ;P17C42.INC Standard Header File, Version 1.03 Microchip Technology, Inc.
                     00264         LIST
                     00018
  00000080           00019 TX_BUFFER       equ     0x80
  000000B0           00020 RX_BUFFER       equ     0xB0
  00000020           00021 RXPTR           equ     0x20
  00000021           00022 TXPTR           equ     0x21
  00000022           00023 SERFLAG         equ     0x22
  00000023           00024 RTINUM          equ     0x23
  00000001           00025 RXDONE          equ     1
  00000000           00026 TXDONE          equ     0
  00000002           00027 HILOB           equ     2
                     00028 ;
                     00029 ;
0000                 00030         ORG     0
0000 C072            00031         goto    start
                     00032 ;
0010                 00033         ORG     0x0010          ;vector for tmr0 interrupt
                     00034 ;tmr0_int                       ;not used here
                     00035 ;
0020                 00036         ORG     0x0020          ;vector for peripheral interrupt
0020                 00037 perf_int
0020 C04D            00038         goto    service_perf    ;service the interrupts
                     00039 ;
0030                 00040         ORG     0x0030
                     00041 ;
                     00042 ;initialize the serial port baud rate interrupts etc.
```

```
0030                00043 init_serial
0030 2922           00044         clrf    SERFLAG, F      ;clear all flags
0031 B800           00045         movlb   0               ;select bank 0
0032 B007           00046         movlw   0x07            ;select 9600 baud
0033 7700           00047         movfp   W,SPBRG         ;       /
0034 B090           00048         movlw   0x90            ;set up serial pins
0035 7300           00049         movfp   W,RCSTA         ;       /
0036 2915           00050         clrf    TXSTA, F        ;setup transmit status
0037 B801           00051         movlb   1               ;select bank 1
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
0038 2916           00052         clrf    PIR, F          ;clear all interrupts
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
0039 2917           00053         clrf    PIE, F          ;clear all enables
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
003A 8017           00054         bsf     PIE,RCIE        ;enable receive interrupt
003B B0B0           00055         movlw   RX_BUFFER       ;set pointer to rx buffer
003C 4020           00056         movpf   W,RXPTR         ;       /
003D 2907           00057         clrf    INTSTA, F       ;clear all interrupts
003E 8307           00058         bsf     INTSTA,PEIE     ;enable peripheral ints
003F 0005           00059         retfie
                    00060 ;
                    00061 ;start transmission of first two bytes
0040                00062 start_xmit
0040 B800           00063         movlb   0               ;select bank 0
0041 8515           00064         bsf     TXSTA,TXEN      ;enable transmit
0042 AB00           00065         tablrd  1,1,W           ;load latch
0043 A216           00066         tlrd    1,TXREG         ;load high byte
0044 B801           00067         movlb   1               ;select bank 1
0045                00068 empty_chk
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
0045 9116           00069         btfss   PIR,TXIF        ;TXBUF empty?
0046 C045           00070         goto    empty_chk       ;no then keep checking
0047 B800           00071         movlb   0               ;select bank 0
0048 A916           00072         tablrd  0,1,TXREG       ;load lo byte
0049 B801           00073         movlb   1               ;select bank 1
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
004A 8117           00074         bsf     PIE,TXIE        ;enable transmit interrupts
004B 8222           00075         bsf     SERFLAG,HILOB   ;set up next for high byte
004C 0002           00076         return
                    00077 ;
                    00078 ;
                    00079         PAGE
                    00080 ;
004D                00081 service_perf
                    00082 ;check for transmit or receive interrupts only
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
004D 9816           00083         btfsc   PIR,RCIF        ;RX buffer full?
004E C062           00084         goto    service_recv    ;yes then service
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
004F 9116           00085         btfss   PIR,TXIF        ;TX buffer empty?
0050 C060           00086         goto    exit_perf       ;no, ignore other int.
0051                00087 service_xmt
0051 9822           00088         btfsc   SERFLAG,TXDONE  ;all done?
0052 C060           00089         goto    exit_perf       ;yes then quit
0053 9A22           00090         btfsc   SERFLAG,HILOB   ;if clr, do low byte
0054 C057           00091         goto    rd_hi           ;else read high byte
0055 A900           00092         tablrd  0,1,W           ;read lo
0056 C058           00093         goto    sx_cont         ;continue
0057                00094 rd_hi
0057 A200           00095         tlrd    1,W             ;read high byte
0058                00096 sx_cont
0058 3A22           00097         btg     SERFLAG,HILOB   ;toggle flag
0059 B800           00098         movlb   0               ;bsr=0
005A 4016           00099         movpf   W,TXREG         ;load tx reg
005B 3300           00100         tstfsz  W               ;last byte?
005C C060           00101         goto    exit_perf       ;no then cont
```

```
005D                00102 end_xmt                         ;else end transmit
005D B801           00103         movlb   1               ;select bank 1
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
005E 8917           00104         bcf     PIE,TXIE        ;disable tx interrupt
005F 8022           00105         bsf     SERFLAG,TXDONE  ;set done flag
0060                00106 exit_perf
0060 8F07           00107         bcf     INTSTA,PEIF     ;clear peripheral int
0061 0005           00108         retfie
                    00109 ;
0062                00110 service_recv
0062 9922           00111         btfsc   SERFLAG,RXDONE  ;RX complete?
0063 C060           00112         goto    exit_perf       ;exit int
0064 6120           00113         movfp   RXPTR,FSR0      ;get pointer
0065 B800           00114         movlb   0               ;select bank 0
0066 6014           00115         movfp   RCREG,INDF0     ;load received value
0067 290A           00116         clrf    WREG, F         ;clr W
0068 3200           00117         cpfsgt  INDF0           ;value = 0?
0069 C06D           00118         goto    end_recv        ;yes then end
006A 1501           00119         incf    FSR0, F         ;inc pointer
006B 4120           00120         movpf   FSR0,RXPTR      ;save pointer
006C C060           00121         goto    exit_perf       ;return from int
006D                00122 end_recv
006D 8122           00123         bsf     SERFLAG,RXDONE  ;set flag
006E 2907           00124         clrf    INTSTA, F       ;clear all int
006F B801           00125         movlb   1               ;select bank 1
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
0070 8817           00126         bcf     PIE,RCIE        ;disable rx interrupts
0071 C060           00127         goto    exit_perf       ;return
                    00128         PAGE
                    00129
                    00130 ;
0072                00131 start
0072 2909           00132         clrf    FSR1, F         ;assign FSR1 as S.P.
0073 0709           00133         decf    FSR1, F         ;     /
0074 B020           00134         movlw   0x20            ;clear ram space
0075 6100           00135         movfp   W,FSR0          ;do indirect addressing
0076                00136 start1
0076 2900           00137         clrf    INDF0, F        ;clear ram
0077 1F01           00138         incfsz  FSR0, F         ;inc and skip if done
0078 C076           00139         goto    start1
0079 E030           00140         call    init_serial     ;initialize serial port
007A B000           00141         movlw   LOW   MESSAGE   ;load table pointer
007B 400D           00142         movpf   W,TBLPTRL       ;    /
007C B001           00143         movlw   HIGH  MESSAGE   ;    /
007D 400E           00144         movpf   W,TBLPTRH       ;    /
007E E040           00145         call    start_xmit      ;start transmission
007F                00146 chk_end
007F 9122           00147         btfss   SERFLAG,RXDONE  ;receive all?
0080 C07F           00148         goto    chk_end         ;no then keep checking
                    00149 ;
0081 C081           00150 loop    goto    loop            ;spin wheel
                    00151 ;
0100                00152         ORG     0x100
0100                00153 MESSAGE
0100 5468 6520 636F 00154         DATA    “The code is: Tea for the Tillerman”
     6465 2069 733A
     2054 6561 2066
     6F72 2074 6865
     2054 696C 6C65
     726D 616E
0111 0000           00155         DATA    0
                    00156 ;
                    00157 ;
                    00158         END
```

```
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : X--------------- ---------------- X--------------- XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XX-------------- ---------------- ---------------- ----------------
0100 : XXXXXXXXXXXXXXXX XX-------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:   102


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     9 reported,     0 suppressed
```

PARITY GENERATION

Since the serial port of the PIC17C42 does not have an on-chip parity generator, parity is generated using software. It takes only 10 program memory words and executes in 10 instruction cycles to generate parity. Since the serial port of the PIC17C42 can operate in a 9-bit mode, the parity bit can be generated in software and transmitted as the 9th bit or it can be compared with the received 9th bit.

In case of transmission, set TX9 to 1 (TXSTA < 6>) to enable 9-bit transmission and write the computed parity bit to TX9D (TXSTA<0>). The 9th bit (parity bit) must be written prior to writing the 8 data bits to TXREG.

In case of a reception, first of all enable 9-bit reception by setting RX9 to 1 (RCSTA<6>). Upon successful reception, the 9th bit is received in RX9D (RCSTA<0>). Parity of the 8 bits of received data is computed using the routine listed below and compared with the 9-bit received.

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

EXAMPLE 6: PRIORITY GENERATION

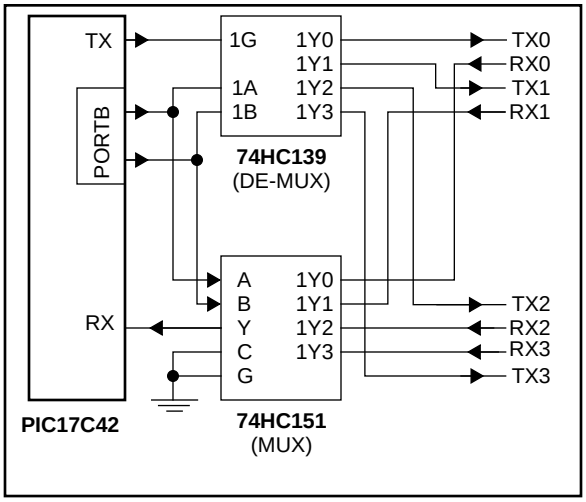
```
MPASM 01.40 Released           PARITY.ASM   1-22-1997  10:17:25         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001         TITLE   “Generate Parity Bit”
                     00002 ;
                    00003 ;*****************************************************************
                     00004 ;       Generate Parity Bit for the 8 bit register ‘txmt’
                     00005 ;       The parity bit is stored in Bit 0 of ‘parity’
                     00006 ;
                     00007 ;
                     00008 ;       Program:          PARITY.ASM
                     00009 ;       Revision Date:
                     00010 ;                         1-22-97  Compatibility with MPASMWIN 1.40
                     00011 ;
                    00012 ;*****************************************************************
                     00013 ;
                     00014         LIST            P = 17C42
                     00015
                     00016         #include        <p17c42.inc>
                     00001         LIST
                    00002 ;P17C42.INC Standard Header File, Version 1.03 Microchip Technology, Inc.
                     00264         LIST
                     00017
                     00018         #define TRUE    1
                     00019         #define FALSE   0
                     00020
                     00021 #define ODD_PARITY      FALSE
                     00022 ;
                     00023
                     00024         CBLOCK  0x20
  00000020           00025                 txmt, parity
                     00026         ENDC
                     00027
0000 1C20            00028         swapf   txmt,W
0001 0C20            00029         xorwf   txmt,W
0002 0121            00030         movwf   parity
0003 2121            00031         rrncf   parity, F
0004 2121            00032         rrncf   parity, F
0005 0C21            00033         xorwf   parity,W
0006 B503            00034         andlw   0x03
0007 B101            00035         addlw   0x01
0008 210A            00036         rrncf   WREG, F
0009 0121            00037         movwf   parity
```

```
                    00038 ;
                    00039  #if ODD_PARITY
                    00040          btg     parity,0
                    00041  #endif
                    00042 ;
                    00043          END

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXX------ ---------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:    10


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

SERIAL PORT EXPANSION

The PIC17C42 has only one serial port. For applications that require the PIC17C42 to communicate with multiple serial ports, a scheme that multiplexes and demultiplexes the RX and TX pins is provided below. This method is suitable only if no more than one USART is needed at any one time. This is the case in many applications where the microcontroller drives several output devices serially. Figure 1, shown below suggests a way to expand the on-chip serial port to four serial ports. To use the scheme shown in Figure 1, The PIC17C42 must select the desired serial port by appropriately setting the two pins of PORTB. The same scheme may be used to further expand the serial ports by using more I/O Ports.

FIGURE 1: MULTIPLEXING THE ON-CHIP USART



RS-232 INTERFACE

Two circuits are provided to interface the CMOS levels of a PIC17C42 to the RS-232 levels. Figure 2 provides an interface to MAX232 (MAXIM's RS-232 Driver/Receiver) with a single +5V power supply. Figure 3 provides a low cost two-chip solution for the RS-232 level translation using a single +5V supply (Note that V- of MC14C88 is connected to DTR of RS-232 Interface. By asserting DTR to low, V- gets the negative voltage from the RS-232 line). An alternative single chip low cost solution is provided in Figure 4. However, 3V sources (+5, +12, -12) are necessary.

FIGURE 2: RS-232 INTERFACE TO MAX232

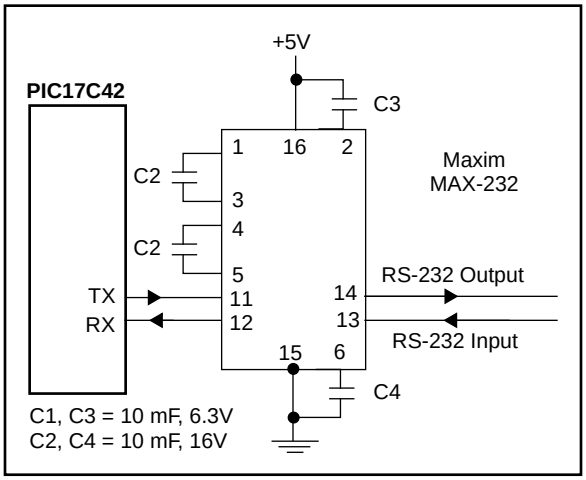


FIGURE 3: LOW-COST TWO-CHIP SOLUTION USING SINGLE POWER SOURCE

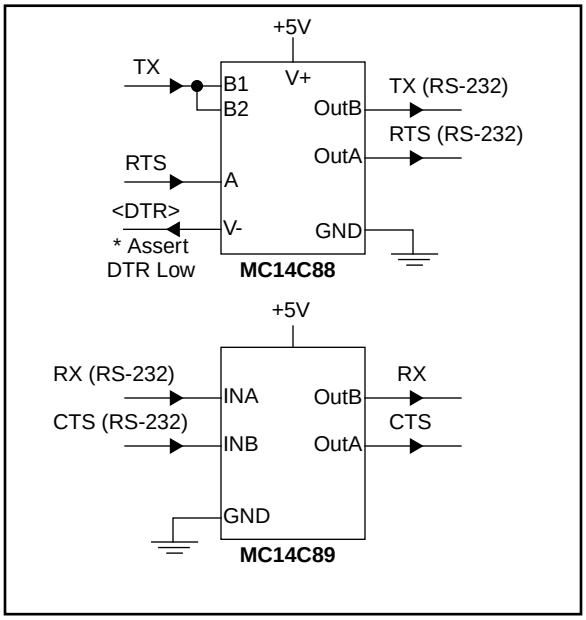


FIGURE 4: LOW-COST SINGLE CHIP SOLUTION USING THREE-POWER SOURCES

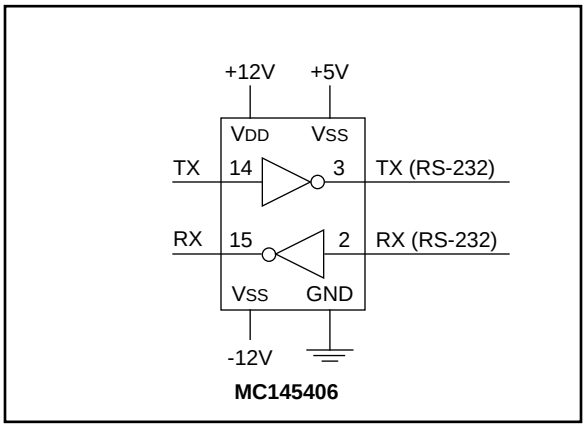


Table 1 provides the summary of RS-232 and V.28 Electrical Specifications.

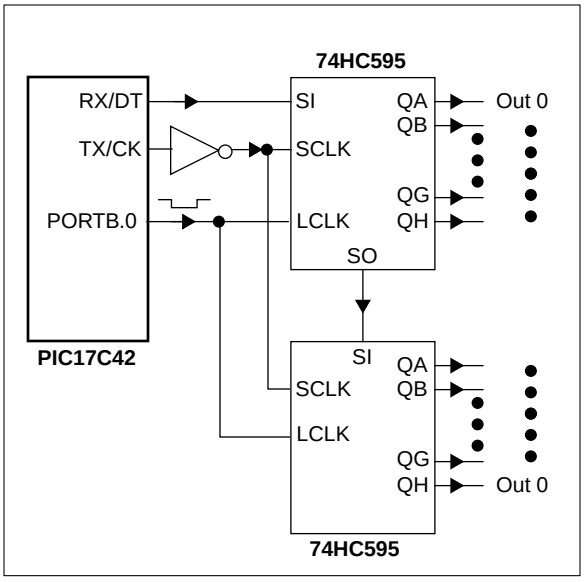
TABLE 1: SUMMARY OF RS-232C AND V.28 ELECTRICAL SPECIFICATIONS

Parameter	Specification	Comments
Driver Output Voltage		
0 level 1 level Max. output	+5V to +15V -5V to -15V ±25V	With 3-7 kΩ load With 3-7 kΩ load No load
Receiver Input Thresholds (Data and clock signals)		
0 level 1 level	+3V to +125V -3V to -25V	
Receiver Thresholds RTS, DSR, DTR		
On level Off level	+3V to +25V Open circuit or -3V to -25V	Detects power Off condition at driver
Receiver Input resistance		
	3 kΩ to 7 kΩ	
Driver Output resistance		
Power off condition Driver slew rate	300Ω Min. 30V/µs max.	V_{OUT} < ±2V 3 kΩ < RL < 7k 0pF < CL < 2500 pF
Signalling rate		
	Up to 20K bits/sec.	
Cable length		
	50'/15m. Recommended max. length.	Longer cables permissible, if C_{LOAD} ≤ 2500 pF

I/O PORT EXPANSION USING SYNCHRONOUS MODE

Although the PIC17C42 has 33 I/O pins, most of these are multiplexed with other peripheral functions. In cases where more I/O ports are needed, the scheme below expands the I/O port using the synchronous mode of serial port by serially shifting the data. Figure 5 shows a scheme to expand the output ports to 16 bits using two standard logic chips (74HC595). The PIC17C42's serial port is configured in synchronous mode and set to be the MASTER. Thus serial data is available on DT (pin 22) and the clock is available on CK (pin 21). The following code will transmit 16 bits serially and clock all the 16 bits at the same time.

FIGURE 5: OUTPUT PORT EXPANSION USING SYNCHRONOUS MODE

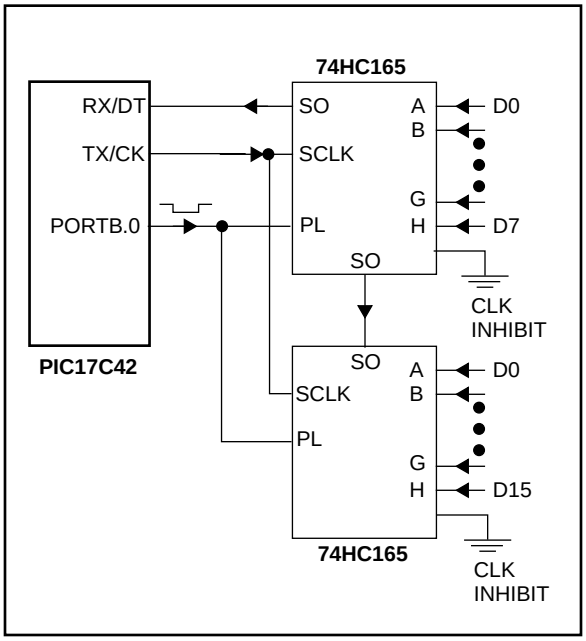


EXAMPLE 7: PORT IMPORT EXPANSION

```
InitSerialPortTxmt
        movlb   0
        clrf    SPBRG                   ; set to highest baud rate = CLKOUT = CLKIN/4
        movlw   0x80
        movwf   RCSTA                   ; enable serial port
        movlw   0xB0
        movwf   TXSTA                   ; 8 bit synchronous master mode
        bcf     DDRB,0                  ; set bit 0 of PortB as output, to be used as Latch Clk
        return
SendSerialData                          ; shift out DataLo & DataHi serially
        movlb   0
        movfp   DataLo,TXREG
        nop                             ; wait for TXREG transfer : if slower baud rate is
                                        ; used check for TBMT reset
        movfp   DataHi,TXREG
wait    btfss   TXSTA,TRMT              ; wait until all 16 bits shifted out
        goto    wait
        bcf     PortB,0
        bsf     PortB,0                 ; clock in the serial data to parallel output of HC595
        return
```

A similar scheme as shown above may be implemented in a reverse manner to expand the input ports. This requires a Parallel In and Serial Out device. Using a standard logic chip (74HC165), the scheme is shown in Figure 6. In order to read the 16-bit input data, the serial port of the PIC17C42 is configured for Synchronous Mode Reception (MASTER mode). An I/O port (PORTB<0>) is used to parallel load the 16 inputs for reading serially. A sample code to read the 16 inputs is shown below.

FIGURE 6: INPUT PORT EXPANSION USING SYNCHRONOUS MODE



EXAMPLE 8:

```
InitSerialPortRcv
           movlb   0
           clrf    SPBRG             ; set to highest baud rate = CLKOUT = CLKIN/4
           movlw   0x80
           movwf   RCSTA             ; enable serial port
           movlw   0x90
           movwf   TXSTA             ; 8-bit synchronous master mode
           bcf     DDRB,0            ; bit 0 of PortB is output, to be used as Parallel
                                     ; Load
           bsf     PortB,0           ; disable parallel Load
           return
ReadSerialData                       ; shift out DataLo & DataHi serially
           movlb   0
           bcf     PortB,0           ; Parallel Load The Inputs into 74HC165
           bsf     PortB,0           ; disable parallel Load
           bsf     RCSTA,SREN        ; enable single byte reception and wait for data
           movlb   1
   wait1   btfss   PIR,RBIF
           goto    wait1             ; check until 8-bits are received
           movlb   0
           movpf   RCREG,DataLo      ; 1st byte is read
           movlb   0
           bsf     RCSTA,SREN        ; enable another byte of reception and wait for data
           movlb   1
   wait2   btfss   PIR,RBIF
           goto    wait2             ; check until 8-bits are received
           movlb   0
           movpf   RCREG,DataHi      ; 2nd byte is read
           return
```

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-786-7200 Fax: 480-786-7277
Technical Support: 480-786-7627
Web Address: http://www.microchip.com

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
Microchip Technology Inc.
4570 Westgrove Drive, Suite 160
Addison, TX 75248
Tel: 972-818-7423 Fax: 972-818-2924

Dayton
Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit
Microchip Technology Inc.
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

AMERICAS (continued)

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong
Microchip Asia Pacific
Unit 2101, Tower 2
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

Beijing
Microchip Technology, Beijing
Unit 915, 6 Chaoyangmen Bei Dajie
Dong Erhuan Road, Dongcheng District
New China Hong Kong Manhattan Building
Beijing 100027 PRC
Tel: 86-10-85282100 Fax: 86-10-85282104

India
Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Japan
Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222-0033 Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai
Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700 Fax: 86 21-6275-5060

ASIA/PACIFIC (continued)

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5858 Fax: 44-118 921-5835

Denmark
Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

11/15/99

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV — DNV MSC The Netherlands Accredited by the RvA
ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.