1—GENERAL OVERVIEW

The GTEK PCSS–8 and the PCSS–4 Super Serial Cards are flexible and powerful serial enhancement products for the IBM Personal Computer (PC, XT, AT, and PS2 model 30) family. Any time the PCSS–8 is mentioned, the same thing applies to the PCSS–4 except for the difference in the number of ports and some I/O addresses. Just substitute 4 for 8 in this manual if you have a PCSS–4. The same thing applies to the PCSS–4T and other models, just add the “T” into the part number next to the number. The “T” means that it has a telephone type modular connector instead of DB–25’s on a bracket.

The PCSS–8 provides 8 independent IBM compatible RS–232 asynchronous serial communication channels per board. Up to four standard PCSS–8 boards may be used for a total of 32 RS–232 channels. Additional boards with custom I/O addresses may be ordered if necessary.

The Model PCSS–nxx can be ordered several different ways. This manual covers three different models, the PCSS–nA, the PCSS–nU and the PCSS–nX. The little “n” stands for either 8 or 4 in the model number. The PCSS–8A and PCSS–4A always uses what we call the DOS Compatible Mode (DCM). The PCSS–8U and PCSS–4U always uses what we call the I/O mapping Mode (IOM). The PCSS–8X and the PCSS–4X can use either of the 2 modes. It can be switched under program control if it is jumpered for what we call the I/O mapping Mode (IOM).

The PCSS–8X and PCSS–8A are supplied with several valuable utility programs: SS.COM allows you to switch the PCSS–nA boards from the DOS comand prompter. SS can also switch the PCSS–nX in either the DCM or IOM modes. On both SS physically maps the desired channel to the standard COMx: I/O addresses and interrupts. SS may be used on the DOS command line, or in batch files, or with the shell command from within BASIC. RES14.COM is a “Terminate and Stay Resident” utility program that **enhances** the bios INT14 function call when using the DOS compatibility mode. SS.COM and RES14.COM will NOT work with the I/O mapping method.

The PCSS–8X can be configured as either DOS compatible or an I/O mapping type of board under program control. The IOM may be used with existing application software which knows how to access multiple uarts.

This is a list of the different models covered in this manual:

DCM	Dos Compatibility Mode	IOM	I/O mapping Mode
n	either 4 or 8 channels	A	capable to 8MHz Bus I/O
B	capable of up to 16MHz Bus I/O operations	C	as previous + FIFO
D	Digiboard compatible	x	stands for : A, B, C, D, X or U

PCSS–nxx, –nTxx, –nDx Means any of the below models

PCSS–4xx or –4Txx 4 Channel board, various configurations as below:

PCSS–4	DCM only, 8MHz	PCSS–4A	same as above (8250)
PCSS–4B	DCM only, 16MHz (16450)	PCSS–4C	DCM only, 16MHz, (16550–fifo)
PCSS–4X	DCM or IOM, 8MHz	PCSS–4XA	same as above
PCSS–4XB	DCM or IOM, 16MHz	PCSS–4XC	DCM or IOM, 16MHz (16550)
PCSS–4U	IOM only, 8MHz	PCSS–4UA	same as above
PCSS–4UB	IOM only, 16MHz	PCSS–4UC	IOM only, 16MHz (16550)

PCSS–8xx or –8Txx 8 Channel board, various configurations as below:

PCSS–8	DCM only, 8MHz	PCSS–8A	same as above
PCSS–8B	DCM only, 16MHz	PCSS–8C	DCM only, 16MHz (16550)
PCSS–8X	DCM or IOM, 8MHz	PCSS–8XA	same as above
PCSS–8XB	DCM or IOM, 16MHz	PCSS–8XC	DCM or IOM, 16MHZ (16550)
PCSS–8U	IOM only, 8MHz	PCSS–8UA	same as above
PCSS–8UB	IOM only, 16MHz	PCSS–8UC	IOM only, 16MHz (16550)

PCSS–8Dx or PCSS–8TDx 8 channel board:

PCSS–8D	Digiboard, 8MHz	PCSS–8DA	Digiboard, 8HMz
PCSS–8DB	Digiboard, 16MHz	PCSS–8DC	Digiboard, 16MHz FIFO

Example Model #'s PCSS–8TXA, PCSS–8XA, PCSS–4X, PCSS–8TUC.

2—COMPATIBILITY

The PCSS–nxx is completely compatible with all existing IBM PC/XT/AT models. It is also compatible with the Compaq and IBM look alikes. The standard PCSS–nxA may be used with PC's, XT's, and AT's with clock frequencies up to 8 MHz (up to 8MHz I/O Bus). The PCSS–nxB may be used with clocks as high as 20 Mhz (up to 20MHz I/O Bus). The PCSS–nxC is the same as the nxB, but uses 16550s.

The most unique feature of the PCSS–nxx is that each of its channels are "compatible" with DOS. PC–DOS has only two logical asynchronous communication channels (COM1: and COM2:). Most application packages may only be installed for one of these two devices.

A problem arises when several application programs involving many serial devices are run on the same computer. The problem is further compounded by the fact that many application programs address the uart directly and use them in an interrupt driven mode. This requires that only "standard" COM1: and COM2: I/O addresses and interrupt request lines be used. The PCSS–nxx board solves this problem.

The PCSS–nXx (in the DOS compatibility mode) addresses and effectively solves this problem by mapping in the selected channel to the standard DOS compatible I/O address and interrupt. That single channel (channel 0–7 or 0–3) can be used with any DOS program that requires a single serial port.

The PCSS–nXx (in the I/O mapping mode) maps each channel to a separate I/O group of addresses. This could be any 4 or 8 contiguous I/O groups (8 x 8 addresses = 64 I/O locations or 4 x 8 = 32) such as 2C0H through 2FFH or 1C0H through 1FFH. This board is compatible with the many other serial boards, application software and operating systems available. An added feature is the ability to know which channel generated an interrupt by reading the scratch register of the highest addressed uart. This is faster than polling each channel to discover which one generated the interrupt.

The PCSS–nXx can switch mode to the DCM mode by writing a 1 to the scratch register's bit 3, if the default method was set to the IOM mode. If the default set was the DCM mode then SS.COM cannot switch the PCSS–nX mode.

The DOS compatibility mode will enable you to write applications faster using the RES14.COM driver and the SS.COM program. Or you can use applications already written for I/O mapping or write them yourself.

A high quality chassis bracket attaches to the rear panel of the computer and is provided with the model PCSS–8. The bracket houses eight (or 4) IBM compatible DB–25 male connectors configured as Data Terminal Equipment (DTE).

The Models that have a "T" in the part number, such as PCSS–8TA or PCSS–8TX are identical to the PCSS–8A and 8TX counterparts, except for the I/O connector. The "T" models use a molded plastic RJ–12 connector that contains all 8 channels and fits into one "slot" on the PC. There is no external bracket involved.

3—INSTALLATION

The PCSS–nxx card can be inserted into any available expansion slot in a PC, XT or AT type computer, provided it can hold a two thirds size card or longer.

CAUTION

Be sure that the power is off and the power cord is removed from the PC before installing or removing any equipment.

STEP 1 – PCSS–nxx I/O and IRQ CONFIGURATION

The PCSS–nxx is shipped from the factory configured as COM2: using interrupt request line IRQ3. If COM1: is desired see the appendix tables for the 4 or 8 channel or “T” configuration cards.

If you need to change the configuration of your board, now is the time to do it. Refer to the tables in Appendix D to change the configuration of your particular model.

PCSS–8U, PCSS–4U has no COM1: compatibility. Channel 7 or Channel 3 is COM2: compatible when this board is shipped. See Appendix D for the I/O channels selected.

Interrupts 5, 7, and 2 are also available on JB1 for custom installations. JB3 on an 8 (or 4) or JB2 on an8T (or 4T) allows two board base addresses thereby allowing as many as 32 channels with standard PCSS–nxx boards. If it is desired to have more than 32 channels, a custom set of board base addresses may be obtained by using a custom address decode PAL available from GTEK. See Appendix D for jumper configurations.

STEP 2 – DB CONNECTOR BRACKET INSTALLATION.

The chassis DB bracket is designed to fit either a PC/XT or AT. Model 30 users can contact GTEK for PS/2 bracket availabilty. If you have what we call a “T” board such as a PCSS–8TX, go to step 3.

For installation on a PC or XT:
(see figure 1) There are 2 ribbon cables which connect the DB bracket to the PCSS–8xx card and 1 for the PCSS–4xx. One is longer than the other. The shorter of the 2 connects to the header for channels 0-3 on the DB bracket (Look at the front of the bracket). They should be folded so that they exit the bracket through the top. Be sure to take note which side you attach to pin 1 of the DB bracket header, since that same side of the ribbon must connect to pin 1 of the header on the PCSS–8xx or PCSS–4xx board.

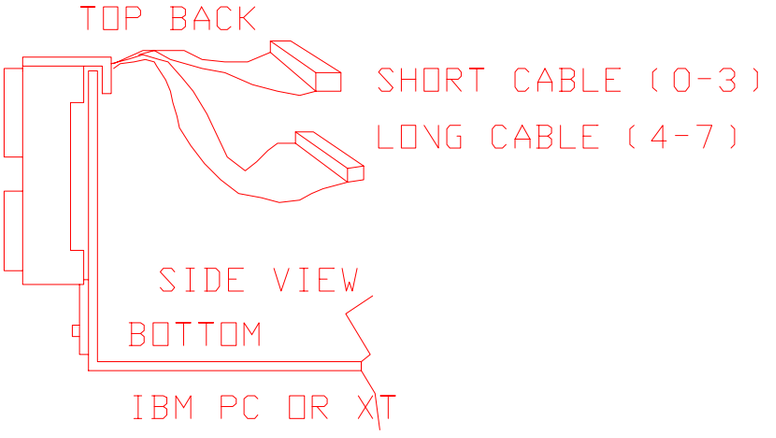


Figure 3.1

The DB bracket is vented and installs over the power supply fan. Care should be exercised so that the cables are folded tightly and kept next to the printed circuit boards that hold the DB connectors so that they will not impede the air flow from the power supply fan.

Remove the 1" standoff from the DB bracket since it's used to secure the DB bracket on an AT and is not used for PC installation. Remove the power supply screw on the computer's back panel. This screw will be used to secure the bottom of the DB bracket to the back of the PC.

Slip the DB bracket into place over the back panel and secure it and the equipment ground wire with the power supply screw. The ground wire attaches to pin 1 of all 8 (or 4) DB connectors. It is for your protection and your computer's protection, so use it! Go to STEP 3.

For installation on a AT; (see figure 2)
There are 2 ribbon cables which connect the DB bracket to the PCSS–nxx card. One is longer than the other. The shorter of the 2 connects to the header for channels 0-3 on the DB bracket (Look at the front of the bracket). Remove the small (1x2") access plate in the

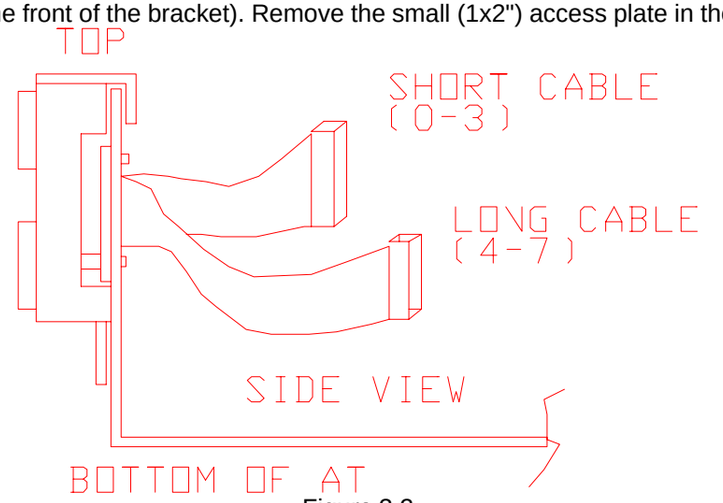


Figure 3.2

middle of the AT back panel. This makes an excellent place to route the ribbon cables into the computer. Attach the ribbon cables to the DB bracket. Be sure to take note which side of the ribbon you attach to pin 1 of the DB bracket header, since that same side must connect

to pin 1 of the header on the PCSS–8 board. Feed the ribbon cables through the opening in the rear panel and attach the bracket to the upper lip of the rear panel.

Use the plate (1"x 2") you removed to secure the DB bracket to the computer. This can be done by bolting the plate to the 1 inch standoff on the bracket from inside the computer, just below the ribbon cables.

Attach the equipment ground wire from the bracket to the rear panel of the AT using the power supply screw below and to the left of the bracket. The ground wire attaches to pin 1 of all eight DB connectors. It is for your protection and your computer's protection, so use it!

STEP 3 – BOARD INSTALLATION

Select an open expansion slot. Locate the metal bracket that covers the cut–out in the back panel for the slot you've selected. Remove the bracket–retaining screw using a small flat head screwdriver. Remove the bracket. Save the screw.

Route the ribbon cables over or through the chassis and connect them to the board making sure that the cable servicing channels 0-3 goes to the header closest to the rear of the computer. Also make sure that the ribbons are attached so that pin 1 on the PCSS–nxx board headers go to pin 1 on the DB bracket headers. It won't hurt anything if you get it wrong, but it won't work either.

(OPTIONAL 422–4 [or 422–2 on an 8T] board)

The model 422–4 is an optional add–on board that plugs on top of a PCSS–nxx board. It gives you the capability of converting RS–232 to RS–422, 4 channels at a time. You could have 4 RS–422 channels and 4 RS–232 channels on a PCSS–8 or 8 RS–422 channels.

The model 422–2 is an optional add–on board that plugs on top of a PCSS–nTxx board. It gives you the capability of converting RS–232 to RS–422, 2 channels at a time. You could have 2 RS–422 channels and 6 RS–232 channels on a PCSS–8T or 4, 6 or 8 RS–422 channels.

Unplug a ribbon cable from the PCSS–nxx and plug it into the 422–4 board. Then plug 422–4 board into the PCSS–nxx where the ribbon was originally hooked. Early models of PCSS–8 and PCSS–4 boards should be sent to GTEK to be installed. Call GTEK for Return Authorization numbers. The PCSS-nXx board does not have to be returned.

On a PCSS–nTxx, install a 422–2 board in place of the second and third chips down from the top (1488 then 1489). The top chip is a 1489, but skip that one, and start at the second chip. This converts 2 channels (top 2 in this case) at a time to RS–422. If you want the next 2 channels to also be RS–422, remove the Next pair of chips (1489, 1488) and install another RS–422–2 board. This will make the next 2 channels RS–422.

The same is true for a RS–485 board (ONLY AVAILABLE FOR A “T” card). Remove 2nd and 3rd chips down from top (1488, 1489) and install the RS–485 board in those 2 sockets.

Keeping the top edge of the PCSS–nxx board level, lower the card until its edge connector is resting on the expansion slot receptacle. Using an evenly distributed pressure, press the PCSS–nxx straight down until it seats in the expansion slot. Install the bracket-retaining screw (that was removed at the beginning of STEP 3) to secure the PCSS–nxx bracket to the rear of the PC chassis.

STEP 4 – REINSTALL COVER

Replace the system unit cover by carefully sliding the cover over the chassis from the front until it stops securely against the rear panel. Be careful not to catch the ribbon cables when installing (or removing) the cover. Reinstall the four corner screws you removed earlier to secure the system cover.

Look behind the DB bracket through the air vent to make sure that the air exhaust vent is not blocked by ribbon cables. Do not operate the computer if it is.

STEP 5 – REINSTALL CONNECTORS

Replace the power cord to the system unit and be sure that the keyboard and the monitor connectors are plugged in.

4—OPERATING INSTRUCTIONS

4.1 GETTING STARTED

Channel 0 of a –4, –4X, –8, –8X is the active channel when you turn your computer on. On the –8X or –4X board, you must have the DOS compatibility mode (JB3=OPEN) set. COM2: is compatible with channel 3 on a –4Ux (or a –4Xx in the IOM mode) and channel 7 on a –8Ux (or a –8Xx in the IOM mode).

Channel 7 of the PCSS–8X (Channel 3 of PCSS–4X) is the active port as COM2: when you turn on the computer if you have the I/O mapping mode default selected by JB2=short and JB3=short.

4.1.1 USING SS.COM

SS.COM is compatible with the PCSS–4, –8, –4X, –8X. It is not compatible with the PCSS–4U or –8U boards at all. Switching to different channel numbers on your COM line is easily accomplished by several methods. The easiest way is to use the supplied SS.COM utility program. When executing the SS utility, two parameters must be supplied: the COM port and the channel number. The COM port number tells SS the I/O address where the PCSS–nxx (–nX when in the DOS compatibilty mode) is located while the channel number is the channel you wish to activate.

Example:

```
SS 27
```

In the example above, channel 7 becomes COM2: on the PCSS–nxx board which was previously installed as COM2.

```
SS 14
```

In the example above, if there is a PCSS–nxx installed as COM1:, then channel 4 is activated. If there is no PCSS–nxx installed as COM1:, then this would have no effect whatsoever.

The SS utility may be executed from within other programs such as BASIC. Within basic one could use the SHELL command. You might have a problem using the SHELL command with 2.x DOS and BASIC 3.0. DOS 3.1 works fine.

Example:

```
SHELL "SS 27"
```

The above example selects channel 7 on COM2:.

The way that SS selects the desired port on the PCSS–nxx is to write the desired port number to the I/O address normally occupied by the uart "scratch register". This register is probably never used by application programs. It is not even addressable on all async cards. The "scratch register" is always at the board base address + 7. That means that on a board installed for COM2:, this register is addressed at 2ffh. On COM1: it is addressed at 3ffh.

A faster way to select channel 7 on COM2: (what we refer to as COM27:) is to use the BASIC "OUT" command.

Example:

```
OUT &h2ff,15
```

The above example has the effect of sending a byte of 15 (8+7) to the board addressed at COM2:. This will cause the PCSS–nxx board to select channel 7. A –nXx board would also switch to the DCM mode and select channel 7.

```
OUT &h3ff,&H4
```

The above example has the effect of sending a byte of 0CH to the PCSS–nA or PCSS–nB board addressed at COM1:. This command is not compatible with the PCSS–nX board in the IOM mode. SS can't switch a –nX board to COM1: when in the IOM mode, because the –nX board can't be set to the 380–3F8h range of addresses.

4.2 ADVANCED USAGE

Custom applications may require several channels to be used simultaneously. There are two distinct ways to do this, either polled or interrupt driven. The polled method is simpler but requires more CPU overhead.

4.2.1 POLLED USAGE

There are two ways in which the PCSS–nxx may be used in a polled mode. The first way is not recommended due to the higher CPU overhead involved.

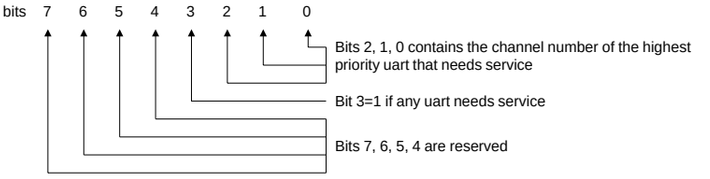
Method A
1—Select uart # 0
2—Read the uart status register to determine if service is required.
3—Select uart # 1
4—Read the uart status register to determine if service is required.
5—Select uart # 2
6—Read the uart status ....
. . . .

16–Select uart # 7
17–Read the uart status register to determine if service is required.

This method might not be too bad if you are only using 1 or 2 of the uarts, but fortunately there is an easier way.

Method B

The board scratch register can be read to determine which, if any, uart requires service. When you read the scratch register, you get the following information:



Channel 0 has the highest priority and Channel 7 has the lowest priority.

To use multiple ports in the polled mode, the scratch register must be periodically polled to determine if any uart requires “service”. By “service” we mean that the uart has received a character. The uart is capable of asking for service for other reasons, also, such as receiver overrun, break detected, etc.

Before you start polling, you need to initialize the uarts to specify which uart will be considered and for what reasons a uart may request service. The PCSS–4X (–8X) board also needs to be in the DOS compatibility mode.

The Interrupt Enable Register at the board address + 1, (2F9h for COM2:), allows you to tell the uart what it may request service for. Bit 0 is of particular interest in that it allows the uart to request service after a character has been received. The following examples, in both basic and assembly language are examples in which uarts 1, 3, and 6 are polled.

Note that just because the uart is told to issue an interrupt request after receiving a character does not mean an interrupt will be generated. See the interrupt driven method for some more details on what it takes to actually generate a interrupt.

Also if you are using the RES14.COM driver, you should not perform the following action. See the example of setting the Uart registers directly using the RES14 driver. It makes things easier!

```
Example 1 - Assembly Language for COM2: board
;initialization code
init:       mov     dx,2ffh ;point to scratch
    mov     al,1    ;select uart #1
    out     dx,al   ;perform the action.
    jmp     in1     ;time delay for AT
in1:        mov     al,1    ;begin IER
    mov     dx,2f9h ;load pointer for IER
    out     dx,al   ;perform the action.
    jmp     in2     ;AT time delay.
in2:        mov     al,3    ;select port #3.
    mov     dx,2ffh ;load SCRATCH
    out     dx,al   ;perform the action.
    jmp     in3     ;time delay.
in3:        mov     al,1    ;select IER
    mov     dx,2f9h ;load pointer for IER
    out     dx,al   ;perform the action
    jmp     in4     ;time delay
in4:        mov     al,6    ;select uart #6
    mov     dx,2ffh ;scratch register.
    out     dx,al   ;select.
    jmp     in5     ;delay
in5:        mov     al,1    ;enable ier
    mov     dx,2f9h ;load pointer to IER
    out     dx,al   ;perform action.
    jmp     in6     ;time delay.
in6:        ret             ;end of initialization
;———-
```

You must realize that other initialization would also be added, such as the baud rate and setting handshake lines. Using RES14 makes things easier, unless you are using a PCSS–8U or a PCSS–8X board in the I/O mapping mode. RES14 cannot be used with an I/O mapping Mode board.

The following is the polling code to determine if an interrupt needs service if you are not using the RES14 driver. This routine returns the Z flag set if no port needs service. Otherwise, if a port does need service, the routine returns the character received in the AL register and the port number in the AH register.

```
;———-
poll:       mov     dx,2ffh ;select scratch
    in      al,dx   ;read scratch
    test    al,8    ;bit 3 set?
    jz      poll1   ;branch, no
;remember that the bottom three bits
;indicate the highest priority uart needing
;service.
    and     al,7    ;need only 3 bits
    out     al,dx   ;select uart
    mov     ah,al   ;return the port
    jmp     po1     ;time delay
po1:        mov     dx,2f8h ;select data
    in      al,dx   ;get character
poll1:      ret             ;return NZ with the
                    ;received data in the
                    ; al register.
```

Example 2 - Basic coding for COM2:

```
UART.INIT:
OUT &H2FF,1 'select uart #1
OUT &H2F9,1 'enable interrupt on Rec Char.
OUT &H2FF,3 'select uart #3
OUT &H2F9,1 'enable interrupt on Rec Char.
OUT &H2FF,6 'select uart #6
OUT &H2F9,1 'enable interrupt on Rec Char.
RETURN


POLL.ROUTINE:
REM returns CHAR$=NULL if no character is

REM available
REM otherwise CHAR$ = char received. PORTNO is
REM returned as the port number the character
REM  came
REM from.
CHAR$=""            'set CHAR$ to NULL
PORTNO=INP(&H2FF)   'get scratch register
                    'content
IF (PORTNO AND 8)=0 THEN RETURN 'bit 3 is 0
PORTNO=PORTNO AND 7         'clear unused bits
OUT &H2FF,PORTNO    'select scratch reg
CHAR$=CHR$(INP(&H2F8))     'receive  the Char
RETURN
```

Remember to look at using the RES14 driver and its examples. RES14 will not work from interpeted BASIC because of the way BASIC handles the COM channels. It has its own drivers instead of using the DOS drivers like it should have. QuickBasic works fine with RES14, as well as many other compiled languages like Turbo–Basic, Turbo–C, Microsoft–C and many others.

4.2.2 INTERRUPT DRIVEN USAGE

Selection of the uart that requires service is similar to polled usage, where the scratch register is read and written to select the uart that requires service, but differs in that you don't need to check bit 3 of the scratch register. This is because you would not be in an interrupt service routine if one of the ports did not need service. RES14 can provide interrupt driven I/O from QuickBasic and other basics. There is no driver provided for interpeted basic, but it is similar to the QB driver.

Interrupt driven usage is beyond the scope of Basic (in our opinion) so only an assembly language example is given. Note that the example is by no means complete and is only given to show some of the steps necessary, relative to the PCSS–8X in the DOS compatibility mode.

Before an interrupt can be actually generated by the PCSS–nxx, the OUT2 bit in the MODEM CONTROL register must be set. This conforms with the standard IBM async card design. You must set the OUT2 line on any one uart to enable the interrupt line driving hardware for the whole board. It does not matter which one you set, just that one or more is set. The capacity of an individual uart to request an interrupt is controlled by the INTERRUPT ENABLE register (as opposed to the MODEM CONTROL register).

Note also that the 8259 interrupt controller IC on the system board must be programmed before an interrupt will be generated, however, that is beyond the scope of this example. Use the RES14 driver instead.

If we use the same uarts as the polled example previously discussed, then the interrupt generating capability can be enabled by the following;

```
initir:    call   init
;call the same initialization routine as polled mode.
;example to set the Interrupt Enable Registers up.
    mov    dx,2ffh ;scratch register for board
    mov    al,0    ;select port #0
    out    dx,al   ;perform
    jmp    intr1   ;delay for AT
intr1:     mov    dx,2fch ;point to the MCR
    mov    al,0fh  ;set out2, out1, dtr, rts enable
    out    dx,al   ;enable irq hardware
                   ;now initialize interrupt vector.
    push   ds      ;put our comm routine there.
    mov    cx,cs   ;get segment cs to ds
    mov    ds,cx   ;
;   mov    al,0ch  ;for COM1:
    mov    al,0bh  ;for COM2:
    lea    dx,isr  ;set up with interrupt handler.
    mov    ah,25h  ;Install ISR
    int    21h     ;perform
    pop    ds      ;restore segment
                   ;now enable Uart Interrupt
                   ;on Received Character.
    mov    al,1    ;Enable RDR int
    mov    dx,2f9h ;IIR
    out    dx,al   ;
    jmp    xxx1    ;time delay
xxx1:      cli     ;No Interrupts for this next...
    In     al,21h  ;8259 controller port
    and    al,0feh ;Clear IRQ3 mask (COM2:)
;   and    al,0efh ;Clear IRQ4 mask (COM1:)
    out    21h,al  ;perform
    jmp    xxx2    ;time delay
xxx2:      sti            ;Re-enable Interrupts
                   ;remember you must also set up
                   ;queue pointers, program baud
                   ;rate generators line control
                   ;registers, etc.
    ret            ;end of this initialization
dinitir:           ;don't forget to deinitialize
                   ;the board after
                   ;you are done running your
;program, or you might lock the computer up if you receive
another character. Exit through this routine...
```

```
    in      al,21h  ;get mask for 8259
    or      al,10h  ;mask for COM2: off.
    or      al,08h  ;mask for COM1: off.
    21h,al          ;perform
    int     20h     ;all done...
```

Now that you have set the PCSS–nxx up, the board will generate an interrupt whenever uart 1, 3, or 6 receives a character. The following interrupt service routine shows how to handle the interrupt.

```
isr:        ;example interrupt service routine for PCSS–nxx
    pushf           ;save machine status flags
    push    ax      ;save accumulator
    push    bx      ;save pointer
    push    dx      ;ditto
    push    ds      ;the “push all” on the
                    ;80286 is quicker...
    mov     ax,cs   ;get the code segment register
                    ;of this code.
    mov     ds,ax   ;make the data segment
                    ;register the same.
    mov     dx,2ffh;point to the scratch register
                    ;of the PCSS-8
    in      al,dx   ;get scratch data
    and     al,7    ;remove unwanted bits (4-7)
    jmp     isr1    ;time delay for AT
isr1:       out     dx,al   ;select the port indicated
                    ;which is the
    jmp     isr2    ;highest priority uart needing
                    ;service.
isr2:       mov     bl,al   ;port number defines which
                    ;queue to store data into.
    in      al,dx   ;get receive data
    call    storeit ;now store data into queue
                    ;defined by bl
    mov     al,20h  ;handle 8259 interrupt controller
    out     20h,al  ;then continue with:
    pop     ds      ;restore all the registers you
    pop     dx      ;pushed
    pop     bx
    pop     ax
    popf            ;pop all for 80286 is quicker
    retr
    queue1  db ? dup (40)  ;40 byte queue
    queue3  db ? dup (40)  ;40 byte queue
    queue6  db? dup (40)   ;40 byte queue
```

Now, to determine if anything has been received, your program need only examine the queue for the uart of interest to see if any thing is in it. Although using interrupts is complicated, many uarts running at extremely high baud rates may be handled effectively.

The RES14 driver supplied can drive the PCSS–nxx (not the –nUx) board under interrupts with AUTOMATIC handshaking! You should consider using it instead of going directly to the uarts because it can relieve the programmer of a very onerous burden.

RES14 can be used from BASIC, PASCAL, C and other programming languages through the machine language interface. See the Quick-Basic example on the supplied disk.

RES14 in the interrupt driven mode can buffer up to 128 received characters and 128 bytes of history of the MODEM CONTROL register (status). You can also set the levels which RES14 begins handshaking with the DTR and CTS control functions. The queue may also be checked or modified at any time. You can determine if RES14 is present so you can use the enhanced features of INT14 and either warn the user that RES14 is not present or use the normal INT14 function calls.

5—SPECIFICATIONS

All PCSS–nxx boards and PCSS–nTxx
(4 or 8 channel board all versions)

Board dimensions:
PCSS–nxx, 4.2" x 9" (⅔ size card)
(106.7mm x 228.6mm)
PCSS–nTxx, 4.2" x 6.5" (½ size card)
(106.7mm x 165.1mm)

Weight:
PCSS–nxx, 7 ounces
(14.9 oz / ribbons and bracket)
202 grams (422 grams with)
PCSS–nTxx, 7 ounces
202 grams

Power Requirements:
PCSS–8xx 5 volt @ 1.3 amp. max
+12 volt @ 200ma max
—12 volt @ 200ma max
PCSS–8Txx 5 volt @ .6 amp max
+12 volt @ 150 ma max
—12 volt @ 150ma max
PCSS–4xx 5 volt @ .8 amp. max
+12 volt @ 100ma max
—12 volt @ 100ma max
PCSS–4Txx 5 volt @ .3 amp. max
+12 volt @ 60ma typical
—12 volt @ 60 ma typical

Operating Environment: 45–95 deg F.
7–35 deg C.
5% to 95% non-condensing relative humidity.

DB–25P Pin out—Standard IBM DTE
PCSS–nxx

PIN	abbrv.	Name	
1	EG	Equipment Ground	
2	TXD	Transmitted Data	(output)
3	RXD	Received Data	(input)
4	RTS	Request To Send	(output)
5	CTS	Clear To Send	(input)
6	DSR	Data Set Ready	(input)
7	SG	Signal Ground	
8	CD	Carrier Detect	(input)
20	DTR	Data Terminal Ready	(output)
22	RI	Ring Indicator	(input)

Modular RJ–12 Jack on a PCSS–nTxx

NOTE: on a GTEK nTxx card, each modular connector is numbered 1 to 6 from Top (where the mtg. bracket is) to Bottom (where the card edge connector is

Pin#	Function	
1	CTS	Clear To Send (in)
2	TXD	Transmitted Data (out)
3	CD	Carrier Detect (in)
4	GND	Signal/Equipment GrouND (in/out)
5	RXD	Recieved Data (in)
6	DTR	Data Terminal Ready (out)

Model 422-4 RS-422 option, 4 channel add-on board

Board dimensions:
3.8" x 3.1"
(96.5mm x 78.7mm)

Weight:
3 ounces
113.4 grams

Power Requirements:
5 volt @ .360 amp.
(can install 2 on PCSS–8xx)

Operating Environment:
45 - 95 deg F.
(7 - 35 deg C.)
5% to 95% non-condensing relative humidity.

```
Pinout of DB-25 For the RS-422-4 system

PIN      Abbrv.      Name
1   EG   Equipment ground of system
2  +TXD  Transmit Data +
3  +RXD  Received Data +
4  –TXD  Transmit Data –
5  +CTS  Clear To Send +
6  –CTS  Clear To Send –
7   SG   Signal logical Ground of system
8  –RXD  Received Data –
20 +DTR  Data Terminal Ready +
22 –DTR  Data Terminal Ready –

PAIRS:
TXD      2, 4  OUTPUT  data
DTR      20, 22      OUTPUT handshake
RXD      3,8   INPUT data
CTS      5, 6  INPUT handshake
```

Output pins may be tri–stated by use of the RTS line of the Uart. Channel 0 RTS controls tri–state on channels 0 and 2 and channel 1 RTS controls tri–state on channels 1 and 3.

```
Modular connector For the RS-422-2 system
(PCSS–nTxx, pin 1 is next to mtg brkt)

Modular Connector
PIN      Abbrv.      Name
1  –TXD  Transmitted Data Minus –
2  +RXD  Received Data Plus +
3   nc   no connection (RS–232,cd)
4   GND  Ground
5  –RXD  Received Data Minus –
6  +TXD  Transmitted Data Plus +
PAIRS:
TXD      1,6   OUTPUT  data
RXD      2,5   INPUT data
```

On a GTEK card with modular connectors, a “T” card, using a RS–422–2 adapter, Note that you can’t “loopback” by just plugging a “DTE” cable in channel 0 and then the other end into channel 1. The transmit never gets connected to receive in that case. You have to “twist” each pair of wires to make the connection on a DTE cable, so that TXD+ goes to RXD+ and TXD– goes to RXD–. Note there are no handshake lines as on the RS–422–4.

```
Pinout of DB-25 For the RS-485–2 system

Modular connector
PIN      Abbrv.      Name
1        nc          no connection
2        +TX/RX      Transmit/Receive Data +
3        nc          no connection
4        GND         Ground – (maybe not used)
5        –TX/RX      Transmit/Receive Data –
6        nc          no connection
```

On a GTEK card with modular connectors, a “T” card, using a RS–485 adapter, you have to assert DTR to transmit and negate it to receive for each channel. You cannot use a standard “DTE” modular cable to “loopback” one channel to another channel, because TX+ needs to be connected to RX+. TX+ would be connected to RX— using a standard “DTE” modular cable.

6—SOFTWARE LICENSE AGREEMENT

“This software is a proprietary product of GTEK, INC. It is protected by copyright and trade secret laws. It is licensed (not sold) for use on a single microcomputer system, and is licensed only on the condition that you agree to this License Agreement.” GTEK, INC. provides this program and licenses its use worldwide. You assume responsibility for the use of this software to achieve your intended results, and for the installation, use and results obtained from the software.

LICENSE

The Licensee may:
a. use the program on a single machine;
b. copy the program into any machine readable or printed form for backup or modification purposes in support of your use of the program on the single machine;
c. modify the program and/or merge it into another program for your use on the single machine (Any portion of this program merged into another program will continue to be subject to the terms and conditions of this Agreement.): and,
d. transfer the program and license to another party if the other party agrees to accept the terms and conditions of this Agreement. If you transfer the program, you must at the same time either transfer all copies whether in printed or machine readable form to the same party or destroy any copies not transferred; this includes all modifications and portions of the program contained or merged into other programs.

You must reproduce and include the copyright notice on any copy, modification or portion merged into another program.

You May Not Use, Copy, Modify, Or Transfer The Program, Or Any Copy, Modification Or Merged Portion, In Whole Or In Part, Except As Expressly Provided For In This License. If You Transfer Possession Of Any Copy, Modification Or Merged Portion Of The Program To Another Party, Your License Is Automatically Terminated.

TERM

The license is effective until terminated. You may terminate it at any other time by destroying the program together with all copies, modifications and merged portions in any form. It will also terminate upon conditions set forth elsewhere in this Agreement or if you fail to comply with any term or condition of this Agreement. You agree upon such termination to destroy the program with all copies, modifications and merged portions in any form.

7—LIMITED WARRANTY

7.1 HARDWARE

GTEK, INC., warrants to the original purchaser of this GTEK, INC., product that it is to be in good working order for a period of one year from the date of purchase from GTEK, INC., or an authorized GTEK, INC., dealer. Should this product, in GTEK, INC.'s opinion, malfunction during the warranty period, GTEK will, at its option, repair or replace it at no charge, provided that the product has not been subjected to misuse, abuse, or non–GTEK authorized alterations, modifications, and / or repairs.

Products requiring Limited Warranty service during the warranty period should be delivered to GTEK with proof of purchase. If the delivery is by mail, you agree to insure the product or assume the risk of loss or damage in transit. You also agree to prepay the shipping charges to GTEK.

All Express And Implied Warranties For This Product Including, But Not Limited To, The Warranties Of Merchantability And Fitness For A Particular Purpose, Are Limited In Duration To The Above 1 Year Period. Some states do not allow limitations on how long an implied warranty lasts, so the above limitations may not apply to you.

Under No Circumstances Will GTEK, INC. Be Liable In Any Way To The User For Damages, Including Any Lost Profits, Lost Savings, Or Other Incidental Or Consequential Damages Arising Out Of The Use Of, Or Inability To Use, Such Product. Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, so the above limitations or exclusion may not apply to you.

This Warranty Gives You Specific Legal Rights, And You May Also Have Other Rights Which May Vary From State To State.

The limited warranty applies to hardware products only.

7.2 SOFTWARE

This Product Is Not A Consumer Product Within The Meaning Of The Uniform Commercial Code And Applicable State Law. The Program Is Provided “AS IS” Without Warranty Of Any Kind, Either Expressed Or Implied, Including, But Not Limited To The Implied Warranties Of Merchantability And Fitness For A Particular Purpose. The Entire Risk As To The Quality And Performance Of The Program Is With You. Should The Program Prove Defective, You (Not GTEK, INC.) Assume The Entire Cost Of All Necessary Servicing, Repair Or Correction. Some States Do Not Allow The Exclusion Of Implied Warranties, So The Above Exclusion May Not Apply To You. This Warranty Gives You Specific Legal Rights And You May Also Have Other Rights Which Vary From State To State.

GTEK, Inc. does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error free. However, GTEK, Inc. warrants the diskette(s) on which the program is furnished, to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from date of delivery to you as evidenced by a copy of your receipt.

Licensee herein acknowledges that the software licensed hereunder is of the class which inherently cannot be tested against all contingencies by Licensor. Licensee acknowledges Licensee’s obligation to test all programs produced by the licensed software to determine suitability and correctness prior to use.

7.3 LIMITATIONS OF REMEDIES

GTEK, Inc.’s entire liability and your exclusive remedy shall be:
1. the replacement of any diskette(s) not meeting GTEK’s “Limited Warranty” and which is returned to GTEK, Inc. with a copy of your receipt, or
2. if GTEK, Inc. or the dealer is unable to deliver a replacement diskette(s) which is free of defects in materials or workmanship, you may terminate this Agreement by returning the program and your money will be refunded.

In No Event Will GTEK, INC. Be Liable To You For Any Damages, Including Any Lost Profits, Lost Savings Or Other Incidental Or Consequential Damages Arising Out Of The Use Or Inability To Use Such Program Even If GTEK, INC. Has Been Advised Of The Possibility Of Such Damages, Or For Any Claim By Any Other Party.

Some States Do Not Allow The Limitation Or Exclusion Of Liability For Incidental Or Consequential Damages So The Above Limitation Or Exclusion May Not Apply To You.

7.4 GENERAL

You may not substitute, assign or transfer the license or the program except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

This Agreement will be governed by the laws of the State of Mississippi.

Should you have any questions concerning this Agreement, you may contact GTEK, Inc. by writing to:

GTEK, Inc. Sales and Service
P. O. Box 2310
Bay St. Louis, MS 39521–2310

8—SERVICE

For warranty service or non warranty service, contact GTEK, INC. at (601) 467–8048 to obtain an RMA (Return of Material Authorization number). We will need the serial number and date of purchase. Send the PCSS-8, freight prepaid to:

GTEK, INC.
RMA #####
399 Highway 90
Bay St. Louis, MS. 39520

Be sure to include the RMA number on and in the package so we will know what to do with it. Out of warranty service charges are determined on an hourly labor plus materials basis.

9—Uart Programming

PROGRAMMING THE 8250 UART:

Registers Accessible to the Programmer:

Transmit Buffer	xF8	(write)
Receive Buffer	xF8	(read)
Divisor Latch	xF8	(read/write LSB, dlab=1)
Divisor Latch	xF9	(read/write MSB, dlab=1)
Interrupt Enable	xF9	(dlab=0, read/write)
Interrupt ID	xFA	(dlab=0, read/write)
Line Control	xFB	(dlab=0, read/write)
Modem Control	xFC	(dlab=0, read/write)
Line Status	xFD	(dlab=0, read/write)
Modem Status	xFE	(dlab=0, read/write)
Scratch	xFF	(dlab=0 read/write)

(Note that you can't really get to the scratch register on a GTEK board because of the hardware logic involved.)

Transmit buffer at xF8 (write only, DLAB=0):
Bits 0–7 equal your output data bits 0–7.

Receive buffer at xF8 (read only, DLAB=0):
Bits 0–7 equal your input data bits 0–7.

Divisor latch at xF8 (write/read, DLAB=1):
Bits 0–7 equal your byte for the LSB of the baud rate word during a write. During a Read, Bits 0–7 equal the present state of the LSB of the baud rate word.

Divisor latch at xF9 (write/read, DLAB=1):
Bits 0–7 equal your byte for the MSB of the baud rate word during a write. During a Read, Bits 0–7 equal the present state of the MSB of the baud rate word.

To program the baud rate, write the LSB and MSB of the Baud Rate Word to xF8 and xF9 while bit 7 of the Line Control Register is high. To determine the bytes to write to the Divisor Latches, use this Algorithm:

$$BRW = \frac{1{,}843{,}200}{(\text{desired\_baud\_rate} * 16)}$$

Example:

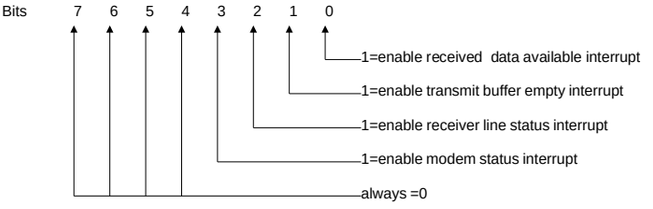
To determine the Baud Rate Word for 9600 baud:
Baud_rate_word = 1,843,200 ÷ (9600 x 16)
Baud_rate_word = 1,843,200 ÷ 153,600
Baud_rate_word = 12
or Baud_rate_word = x000C hex (MSB=00, LSB=0C)

To determine BRW for 300 baud:
BRW = 1,843,200 ÷ (300 x 16)
BRW = 1,843,200 ÷ 4800
BRW = 384
or BRW = x0180 hex (MSB=01, LSB=80)

It looks like the highest baud rate is 115,200 baud (BRW= 0001) and the lowest is 2 baud (BRW= E100). Remember that due to hardware speed limitations, you may not be able to use 115,200 baud or 2 baud. Typically a 6 MHz computer might be able to use 57,600 baud, while an 8 or 12 MHz computer might be able to use 115,200.

Interrupt Enable Register xF9 (write/read, DLAB = 0):

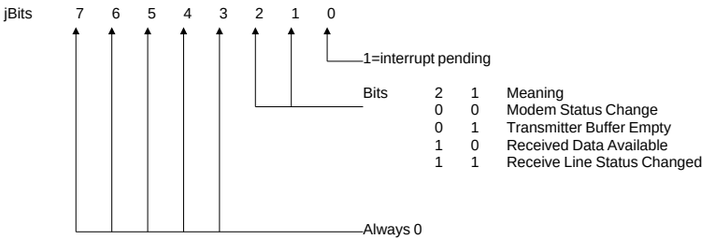
Reading the IER will give you the current state of the bits. Writing to the IER will cause certain things to happen:



Interrupt Identification Register at xFA (read):

Reading the IIR will give you the status of the interrupts, if one occured. This is prioritized, so that if more than 1 interrupt occured, you are vectored to the interrupt service routine with the highest priority. They are prioritized in the following manner:

Priority	Service
1	Receiver Line Status (highest priority)
2	Received Data Ready
3	Transmit Buffer Empty
4	Modem Status Change (lowest priority)



The Modem Status Change (lowest priority- 00) indication is reset by reading the Modem Status Register. This interrupt can be caused by the Clear To Send, Data Set Ready, Ring Indicator, or Received Line Signal Detect (CD) signals.

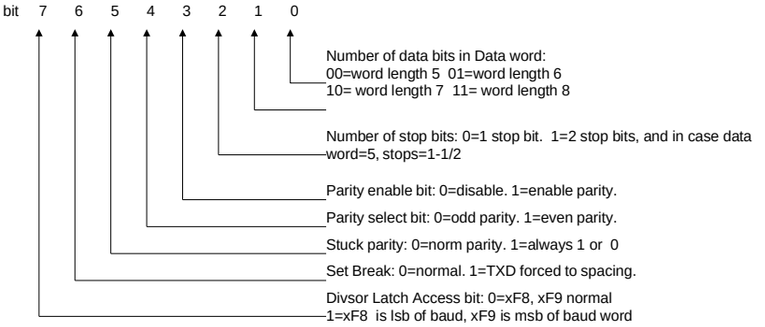
The Transmitter Buffer Empty indication (01). Read the IIR or write to the Transmitter buffer to Reset this Interrupt. This interrupt is caused by the Transmit Buffer becoming Empty.

The Received Data Available indication (10). Read the Receive Buffer Register to Reset this Interrupt. It's caused by the Receive Buffer Register becoming Full.

The Receiver Line Status changed indication (highest priority 11). Read the Line Status Register to reset this Interrupt. It's caused by an Overrun, Parity, or Framing Errors, or the Break Interrupt.

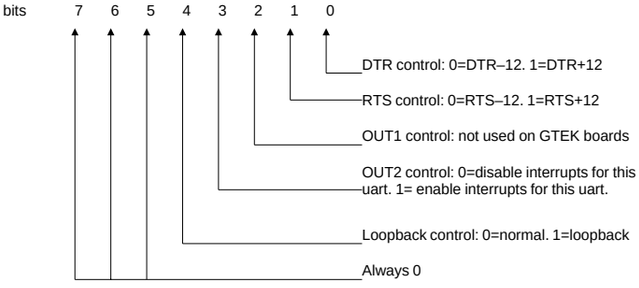
On an interrupt from the uart, the highest priority interrupt has precedence. All other pending interrupts are held until the action that clears the current interrupt is performed.

Line Control Register xFB (read/write):
Reading this register gives you the current settings of the uart as



specified below. Writing to it will change the current settings.

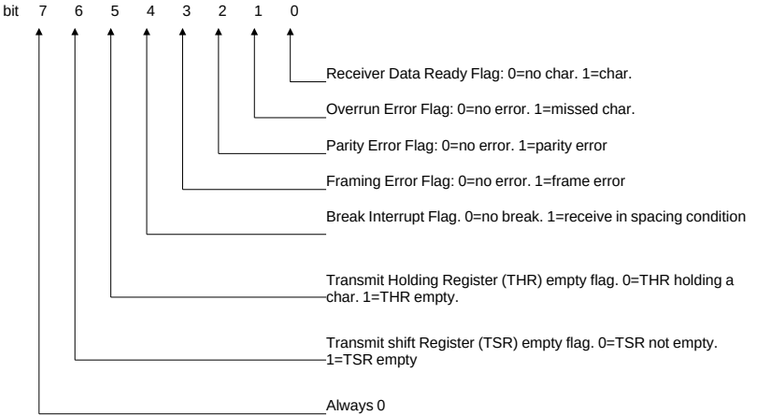
Modem Control Register xFC (write/read):
Reading this register gives you the current settings. Writing this



register controls the desired function.

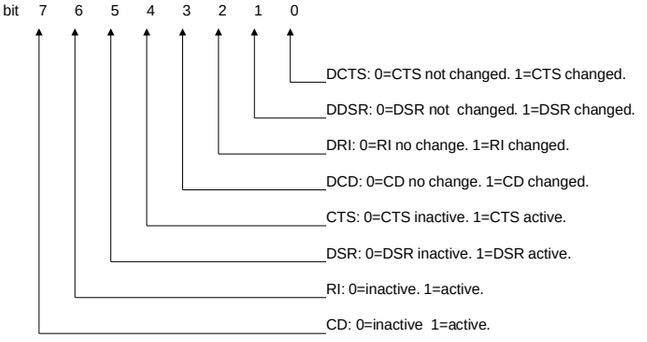
Line Status Register xFD (read/write):

Reading this register will give you the current line status. You read this register in the interrupt mode to find which item caused the interrupt. The interrupt is the highest priority interrupt. You must enable interrupts for these registers to work properly.



Modem Status Register xFE (read/write):

This Register indicates the current state of the external devices (modems, programmers, emulators, terminals).



Suggestion: If you need more information on programming the 8250, then we suggest that you invest in the *IBM TECHNICAL REFERENCE PERSONAL COMPUTER AT* manual. It has more detailed information about programming the uart, and the BIOS routines make a good example.

Technical information and specifications provided in this document are SUBJECT TO CHANGE WITHOUT NOTICE.

10—RS–422 and 485

The OPTIONAL 422–4 RS–422 4 channel board

The model 422–4 is an optional add–on board that plugs on top of a PCSS–nxx board. It gives you the capability of converting RS–232 to RS–422, 4 channels at a time. You could have 4 RS–422 channels and 4 RS–232 channels on a PCSS–8xx or 8 RS–422 channels.

To install in on an existing PCSS–nxx board:

1– Remove the PCSS–nxx from your computer.

2– Unplug the first ribbon cable from the PCSS–nxx and plug it into the 422-4 board.

3– Then plug the 422–4 board into the PCSS–nxx where the ribbon was originally hooked.

4– Re-install the PCSS–nxx and re-route the ribbon cable up and over to the bracket.

Remember that the RS–422 signal output/input levels are NOT compatible with RS–232 and hook differently. The modes are not compatible.

See the specifications section for the wiring of the DB–25 connector. On boot up under DOS, the RTS line for the channel that is DOS compatible (0) will cause 2 transmit channels of the 422–4 to become active (0,2). The other 2 transmit channels will also become active if the RTS line for channel 1 is +12. The receivers are always active.

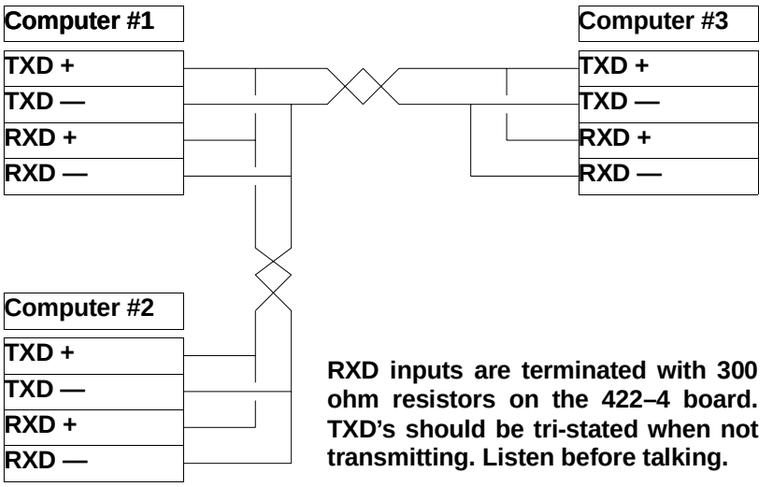
Tri-stating can be controlled by RTS on channel 0 and RTS on channel 1. When DOS or the RES14 driver is controlling the channel, the RTS line is always enabled. If you are controlling it yourself, you must set RTS high (+12) to enable the channel pairs to transmit. RTS is jumpered to DSR on each channel.

Pinout of DB-25 when a Model 422–4 is attached:

PIN #	Abbrv.	Name	Polarity
1	EG	Equipment Ground	none
2	TXD+	Transmit Data	+
3	RXD+	Received Data	+
4	TXD—	Transmit Data	—
5	CTS+	Clear To Send	+
6	CTS—	Clear To Send	—
7	SG	Signal Ground	none
8	RXD—	Received Data	—
20	DTR+	Data Terminal Ready	+
22	DTR—	Data Terminal Ready	—

PAIRS:	TXD	2, 4	Output +,—	Data
	DTR	20, 22	Output+,—	Handshake
	RXD	3, 8	Input+,—	Data
	CTS	5, 6	Input+,—	Handshake

Typical Half–Duplex hookup with 3 computers.



The OPTIONAL 422–2, RS–422 2 channel board

The Model 422–2 is an optional add–on board that plugs on top of a PCSS–nTxx board. It gives you the capability of converting RS–232 to RS–422, 2 channels at a time. You could have 2 RS–422 channels and 6 RS–232 channels on a PCSS–8Txx or 8 RS–422 channels.

To install in on an existing PCSS–nTxx board:

1– Remove the PCSS–nTxx from your computer.

2– Looking at the interface chips from top to bottom next to the modular jacks, the sequence is 1489, 1488, 1489, etc... From the top, remove the 1488, then the following 1489. You can then install a 422–2 board in this location. The sequence then is 1489, 422–2 in 2 sockets, 1488, 1489, etc...

3– If you have no more 422–2s to install, go to step 4. Otherwise, repeat step 2 from the NEXT 1488/89 positions, which will convert the next 2 channels.

4– Re–install the PCSS–nTxx.

Remember that the RS–422 signal output/input levels are NOT compatible with RS–232 and hook differently. The modes are not compatible.

The transmit and receive channels are always enabled on a 422–2, unlike the 422–4.

Modular connector pin–out of a PCSS–nTxx when used with the Model 422–2. (first channel and next channel)

Pin#	Name	Function
1	TXDM	Transmit Minus
2	RXDP	Receive Plus
3	NC	
4	GND	You may or may not use this
5	RXDM	Receive Minus
6	TXDP	Transmit Plus

The OPTIONAL 485–2, RS–485 2 channel board

The Model 485–2 is an optional add–on board that plugs on top of a PCSS–nTxx board. It gives you the capability of converting RS–232 to RS–485, 2 channels at a time. You could have 2 RS–485 channels and 6 RS–232 channels on a PCSS–8Txx or 8 RS–485 channels.

To install in on an existing PCSS–nTxx board, follow the Model 422 directions.

Assert DTR to transmit and negate DTR to receive.

Modular connector pin–out of a PCSS–nTxx when used with the Model 422–2. (first channel and next channel)

Pin#	Name	Function
1	NC	
2	TX/RXDP	Transmit Plus or Receive Plus
3	NC	
4	GND	Signal/Equipment ground
5	TX/RXDM	Transmit Minus or Receive Minus
6	NC	

11— Typical RS–232 HOOK–UPS

The PCSS–8 is a DTE type device. Hook-ups to DCE devices run straight through, pin for pin. 1–1, 2–2, etc.

Modems like Hayes Stack, Novation, etc.

Wires run straight through, 1—1, 2—2, etc.

Serial Printers like Epson MX 100, NEC 7700, Brother, Okidata, and Anadex.

Wires run 1—1, 7—7, 2—3, 3—2, 4—5, 5—11 or 5—19, 6 and 8—20,

Serial Printers like Qume:

Wires run 1—1, 7—7, 2—3, 3—2, 4—5, 5—20, 6 and 8—20

Slow CRT and printers (old):

Wires run 1—1, 7—7, 2—3, 3—2, 4—5, 5—4, 6 and 8—20

Diablo 620 Printer:

Wires run 1—1, 7—7, 2—3, 3—2, 6 and 8—20, 20—6

Diablo 630 printer:

Wires run 1—1, 7—7, 2—3, 3—2, 5—11, 6 and 8—20 or 6—4

MX-80, SC TP-1, IDS:

Wires run 1—1, 7—7, 2—3, 3—2, 4—5, 5—4 on TP-1, 5—20 on MX–80 and IDS, 6 and 8—20 on TP-1 only

Hewlett-Packard, Houston Instruments Plotters:

Wires run 1—1, 7—7, 2—3, 3—2, 5—20

12— Tables

Table 1, PCSS–8 and PCSS–4

JB1	JB2	JB3	address	COM?	SS 8ch	SS 4ch
4	open	open	3F8h	COM1	10–17	10–13
3	short	open	2F8h	COM2	20–27	20–23
?[1]	open	short	3E8h	COM3	30–37	30–33
?[1]	short	short	2E8h	COM4	40–47	40–43

[1]In normal operation, JB1 should be jumpered on 4 for COM1: and 3 for COM2:. You can, however, select any other available IRQ, but unless the application is made to handle that interrupt, it may not work. (3, 4, 5, 7)

Table 2, Model PCSS–8T and PCSS–4T

JB1	JB2	JB3	address	COM?	SS 8ch	SS 4ch
4	open	open	3F8h	COM1	10–17	10–13
3	OPEN	SHORT	2F8h	COM2	20–27	20–23
?[1]	SHORT	OPEN	3E8h	COM3	30–37	30–33
?[1]	short	short	2E8h	COM4	40–47	40–43

[1]In normal operation, JB1 should be jumpered on 4 for COM1: and 3 for COM2:. You can, however, select any other available IRQ, but unless the application is made to handle that interrupt, it may not work. (3, 4, 5, 7)

Table 3, PCSS–8U or PCSS–4U

JB1	JB2	JB3	8 ch address	4 ch address	SS	COM?
?[3]	open	open	280—2BFh	2A0—2BFh	na	na[2]
3[1]	open	short	2C0—2FFh	2E0—2FFh	na	COM2[1]
?[3]	short	open	180—1BFh	1A0—1BFh	na	na[2]
?[3]	short	short	1C0—1FFh	1E0—1FFh	na	na[2]

[1]IRQ 3 should be selected (JB2=open, JB3=short) so you can maintain COM2: compatiblity with channel 7 (or 3) on this particular configuration. You can use whatever IRQ you need to use though. 2, 3, 4, 5, and 7 are selectable.
[2]There is no COM port compatiblity with these selections.
[3]Use any available IRQ selection that will work for the application, 3, 4, 5, or 7.

Table 4, PCSS–8TU or PCSS–4TU

JB1	JB2	JB3	8 ch address	4 ch address	SS	COM?
?[3]	open	open	280—2BFh	2A0—2BFh	na	na[2]
3[1]	SHORT	OPEN	2C0—2FFh	2E0—2FFh	na	COM2[1]
?[3]	OPEN	SHORT	180—1BFh	1A0—1BFh	na	na[2]
?[3]	short	short	1C0—1FFh	1E0—1FFh	na	na[2]

[1]IRQ 3 should be selected (JB2=open, JB3=short) so you can maintain COM2: compatiblity with channel 7 (or 3) on this particular configuration. You can use whatever IRQ you need to use though. 2, 3, 4, 5, and 7 are selectable.
[2]There is no COM port compatiblity with these selections.
[3]Use any available IRQ selection that will work for the application, 3, 4, 5, or 7.

Table 5, Model PCSS–8X or PCSS–4X

JB1	JB2	JB3	8 ch address	4 ch address	SS 8ch (4ch)	COM?
4	open	open	3F8h	3F8h	10–17 (13)	COM1
3	short	open	2F8h	2F8h	20–27 (23)	COM2
?[1]	open	short	280—2BFh	2A0—2BFh	na[2]	na[2]
3[3]	short	short	2C0—2FFh	2E0—2FFh	na[2]	COM2[3]

1 When this configuration is used, use whichever IRQ is available for use.
2 There is no COM port compatibility with this mode
3 To maintain compatibility with COM2:, use IRQ3, then channel 7 on –8X or channel 3 on –4X is DOS compatible on cold boot.

Table 6, Model PCSS–8TX or PCSS–4TX

JB1	JB2	JB3	8 ch address	4 ch address	SS 8ch (4ch)	COM?
4	open	open	3F8h	3F8h	10–17 (13)	COM1
3	OPEN	SHORT	2F8h	2F8h	20–27 (23)	COM2
?[1]	SHORT	OPEN	280—2BFh	2A0—2BFh	na[2]	na[2]
3[3]	short	short	2C0—2FFh	2E0—2FFh	na[2]	COM2[3]

1 When this configuration is used, use whichever IRQ is available for use.
2 There is no COM port compatibility with this mode
3 To maintain compatibility with COM2:, use IRQ3, then channel 7 on –8X or channel 3 on –4X is DOS compatible on cold boot.

Table 7, PCSS–8H

JB1	JB2	JB3	8 ch address		SS	COM?
3	open	open	140–17FH		na	na[1]
3	open	short	200–23FH		na	na[1]
3	short	open	240–27FH		na	na[1]
3	short	short	280–2BFH		na	na[1]

[1] There is no COM port compatiblity with these selections..

Table 8, PCSS–8TH

JB1	JB2	JB3	8 ch address		SS	COM?
3	open	open	140–17FH		na	na[1]
3	short	open	200–23FH		na	na[1]
3	open	short	240–27FH		na	na[1]
3	short	short	280–2BFH		na	na[1]

[1] There is no COM port compatiblity with these selections..

Table 9, PCSS–8D or PCSS–4D

JB1	JB2	JB3	8 ch address		SS	COM?
3	open	open	100–13FH		na	na[1]
3	open	short	148–187H		na	na[1]
3	short	open	188–22FH		na	na[1]
3	short	short	230–26FH		na	na[1]

[1] There is no COM port compatiblity with these selections..

Table 10, PCSS–8TD or PCSS–4TD

JB1	JB2	JB3	8 ch address		SS	COM?
3	open	open	100–13FH		na	na[1]
3	short	open	148–187H		na	na[1]
3	open	short	188–22FH		na	na[1]
3	short	short	230–26FH		na	na[1]

[1] There is no COM port compatiblity with these selections..

The following boards are shipped as:

PCSS–8A, PCSS–4A
JB1=3 (irq3), JB2=short, JB3=open for COM2:, Channel 0 is DOS compatible on cold boot.

PCSS–8AU, PCSS–4AU
JB1=3 (irq3), JB2=open, JB3=short for COM2:, Channel 7 (or 3 on PCSS–4U) is DOS compatible on cold boot.

PCSS–8XA, 4XA, 8TXA, 4TXA
JB1=3 (irq3), JB2=short, JB3=short for PCSS–8U/4U/8TU/4TU emulation, or PCSS–8A/4A/8TA/4TA emulation when override bit is used.

PCSS–8H, 8TH
JB1=3 (irq3), JB2=short, JB3=short for COMTROL (HOSTESS–8) compatibility at 280H.

PCSS–8DA
JB1=3 (irq3), JB2=open, JB3=open for Digiboard compatiblity at 100H.

NOTES:

On all REV–C boards, purchased as a PCSS–8X, JB2 is used for address select and JB3 is used for DOS Compatible Mode / I/O Mapping Mode. To be able to select both modes programatically, you should be using the I/O mapping Mode (JB3=short). Reverse JB2 and JB3 when using a PCSS–8TX board of any kind.

When referring to a 8 channel board (8X, 8TX) Channel 7 would be called the Dos Compatible Channel; Channel 3 is Dos Compatible on a 4 channel board. You may use the board in the Dos Compatible Mode while still in the I/O mapping Mode (JB3=Short). Do this by writing to the scratch register of Channel 7 with bit 3 set to 1 and bits 0–2 set to the channel to select. This causes the selected channel to become the active channel (in the Dos Compatibility Mode). To Return to the I/O Mapping Mode, write to Channel 7 scratch register with Bit 3 set to 0. If you use RES14 and SS, they handle switching automatically.

Bit 3 = 1 of the Channel 7 (8X, 8TX) scratch register will not affect the I/O Mapping Mode except for Channel 7, since data written to Channel 7, (which happens to be a DOS Compatible channel when JB2=Short And JB3=Short, [reverse for 8TX]) will go to the selected channel (including Channel 7, if it is the selected channel).

JB1 selects the DOS interrupt line.

13— RES14.COM

USING RES14.COM

RES14.COM is a serial driver replacement for the PC BIOS int 14h. It allows 16 interrupt driven serial ports using the PCSS–8X (–4X) DOS Compatibility Mode.

The RES14.COM program provides byte stream i/o to the communications ports according to the following parameters:

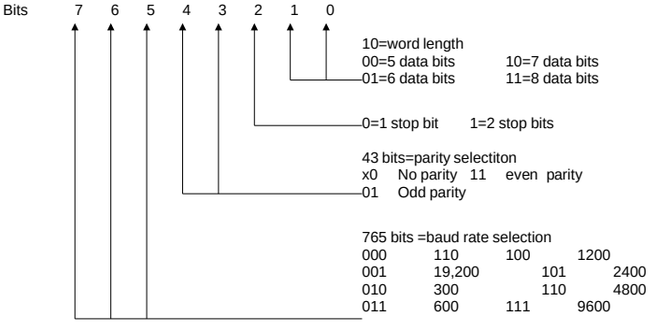
Note: All registers are preserved, except ax.

On entry, DX must contain the port number as follows.
dh = channel no. 0 to 7 (or 0-3 for pcss-4)
dl = comm board 0 or 1, 0 is a com1:, 1 is com2:
ah = the function number.
al = parameters.
On return, AX contains answers (if applicable)

Function Calls for INT 14h
Initialize COM channel(s)

AH = 00H means to initialize the communications port.

AL= nnH means parameters to initialize as follows:



On return from function call 00H, conditions are set in AL the same as if you had made a status function call where AH=03H. Note that function call 00H disables the receive interrupts, so make the function call 00H first, then make the function call 05H (if necessary). RTS and DTR are both set high when the port is initialized.

Transmit A Character

AH = 01H sends the character contained in AL over the selected communications line. AL is preserved.

AL = nnH the character to be sent.

On exit, bit 7 of AH is set if the routine was unable to transmit the byte in AL. If bit 7 is not set, the remainder of AH is set as in a status request (AH=03H), reflecting the current status of the line. Note: The character won't be sent unless the transmit shift register (TSR) is empty AND CTS is high. See function call 0fh.

Receive A Character

AH = 02H receives a character into AL from the communications line before returning to the caller.

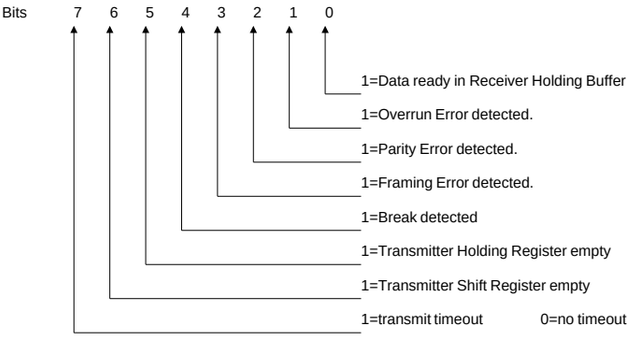
AL = xxH don't care what it is on entry.

On exit, AH contains the current line status, as set by the status routine (AH=03H), except that the only valid bits left are the error bits(7,4,3,2,1). If AH has bit 7=1 (time-out), none of the remaining bits are predictable. Therefore AH is not zero only when an error has occurred.

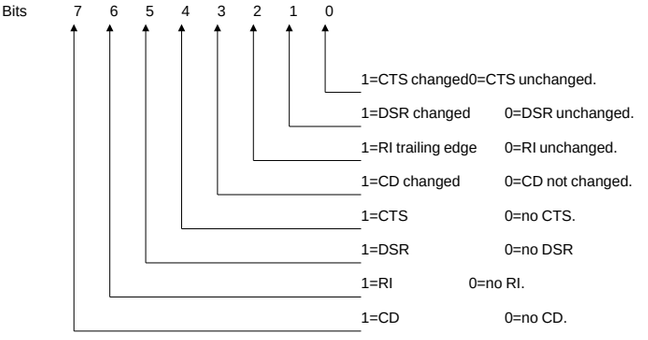
Get Communication Channel Status

AH = 03H returns the communcations port status in the AX register.

AL = xxH don't care what it is on entry.
On exit, AH contains the Line Control Register byte.



When the receive interrupts are enabled, the information for bits 1 through 4 comes from the receive queue. Bit 0 = 1 if there is data in the queue. On exit, AL contains the Modem Status Register byte.



Up to here, the above functions work the same way as the IBM BIOS INT14 function calls.

The following function calls are an enhancement of the INT14 functions.

Check for RES14.COM Installation

AH = 04H check for RES14.COM installed properly.

AL = 00H clear AL for data in case RES14.COM not installed.

On exit, AL = 03H or the current version digit of RES14.COM (03H). AL = 00H if the RES14.COM driver is not present. The DOS error level is set if RES14.COM is installed. It returns 1 if the driver was just installed and a 3 if the driver had been installed previously. The return will be 0 if the installation fails.

Enable/Disable channel Receive Interrupts

AH = 05H enable or disable a channels Interrupt on Received Character.

AL = 00H on entry to disable IRC.

AL = 01H on entry to enable IRC.

On exit, no registers have been modified and the desired function has been performed. Note that the 8259 IRQ channel is not disabled unless all 8 ports on that interrupt line are disabled. Data bytes and MCR bytes are buffered into separate 128 byte queues.

DTR QueueLevel Control

AH = 06H DTR level control function to set the queue levels at which DTR is raised and lowered.

AL = nnH on entry as follows: Bit 7 = 1 to program DTR active level. This is the queue count level at which DTR will go high (+12V) indicating that you are ready to continue receiving data. The default level is 40 characters.

Bit 7 = 0 to program the DTR inactive level. This is the queue count level at which DTR will go low (-12V) to indicate that the device attached to the port should stop sending. If this level is set to an amount larger than the maximum queue size, then DTR will never go low. This may be useful if the port is to be attached to a MODEM, however, it is then left to the user to implement xon/xoff or some other form of flow control. The default level is 80 characters.

Bits 6-0 represent a number which is multiplied by 10 to indicate the level you wish to set. The buffers can hold 128 bytes of data and 128 bytes of historical MCR information each. If you want the DTR to change (to -12V) when to queue fills to 100 bytes then bits 7-0 should be set to 00001010B (0AH). If you wish to raise DTR (to +12V) when the queue empties to 10 bytes then bits 7-0 should be set to 10000001B (81H).

Example: suppose you want DTR to lower (-12V) if the queue fills to 100 characters and to raise (+12V) when the queue level drops to 20 characters. The following code would do the trick. Assume channel 4 on com2:

```
mov     ah,6
mov     al,10     ;10 x 10 = 100 characters
mov     dh,4      ;channel 4
mov     dl,1      ;com2
int     14h       ;do it
mov     al,2      ;10 x 2 = 20
or      al,80h    ;set bit 7 for DTR ON level
int     14h       ;do it
```

Set Transmit Timeout Value

AH = 07H will set the communications Timeout value. The time is indicated in seconds and is only approximate.

AL = nnH on entry for Timeout value in seconds. Note that you probably should not use the Timeout to assume that the channel is not going to transmit (or receive) a

character because it is much faster to just check the status with function call = 03H to see if you can transmit or if there is a character ready to be received. The default Timeout value is approximately 1 second.

Example:

```
lto:    mov     al,33           ;33 second time out
        mov     ah,7            ;function call 7
        mov     dx,0300h        ;channel 3, com1:
        int     14h
        ret
```

RTS Line Level Control

AH = 08H will control the RTS line with the value loaded in AL. This function is provided so that the user has control of RTS, pin 4 on the DB25 connector. RTS is set active (+12V) when the port is initialized, and is additionally set active when a function call 01H (transmit a character), is made.

AL = 00H on entry to set RTS = —12V

AL = 01H on entry to set RTS = +12V. There is nothing returned in AH or AL.

Transmit Break Control

AH = 09H is the Transmit Break Control This function is provided so that the user can send a break (+12V) on the TXD line if he so desires.

AL = 00H on entry to clear the Break (back to -12V).

AL = 01H on entry to set the Break (set to +12V). When set, it remains this way until another function call 09H is made to clear it. Nothing is returned in the registers.

Loopback Mode Control

AH = 0AH Loopback Mode control is provided so that the user can put the channel into the loopback mode for test purposes. Since the control line on OUT2 is fed back to carrier detect, then if you have the channel

IRC enabled, DCD will be 0. If no interrupts are enabled, DCD will be high. RI will remain low since out1 is left low. Nothing is returned in the registers.

AL = 00H on entry sets the normal mode.

AL = 01H on entry sets the Loopback mode.

Request Queue Count

AH = 0BH on entry is the Request Queue Count. It returns the number of characters available in the queue in AL.

AL = xxH on entry, don't care. On return, AL contains the number of characters available in the queue.

Request Queue Flush

AH = 0CH on entry is the Queue Flush. This causes the queue to be emptied of all characters. Flushed, so to speak. It returns nothing in any register and sets DTR to ready (+12V) on return.

DTR Line Level control

AH = 0DH on entry controls the DTR line. This function is provided so that the user has control of DTR, pin 20 on the DB25 connector. This function is different from function 06H in that the line is controlled directly and not by the number of characters in the queue. DTR is set active when the port is initialized, and if no interrupts have been enabled, it is also set high whenever a function call 02H, (Receive character), is made. This function should not be used in the interrupt driven mode since the state of DTR changes according to queue level with that function.

AL = 00H on entry to set DTR to -12V.

AL = 01H on entry to set DTR to +12V.

Get Last Character Received

AH = 0EH on entry will sample the last character received. Returns data and status of the last character received into the queue.

AL = xxH on entry. Returns AX with status in AH and data in AL. You should do function call 0BH first to be sure you are getting valid data. If the queue is empty while this function call is performed, the status and data will not be valid.

This may be useful in sampling for an escape code in real time while there is still data in the queue to be processed. As an example, suppose someone initiated a command on a bulletin board system running with RES14 resident that would cause a large amount of data to be dumped. The user would be able to issue a command that would be immediately recognized by the bulletin board system if this function call was being used to sample characters received instead of waiting until it had to process some characters.

Transmit Character without status check

AH = 0FH will send the character contained in AL over the communications channel without checking any status.

AL = nnH on entry is the character to be transmitted.

AL is preserved and no status is returned. RTS is not affected. This is a primitive transmit function call. It is up to the user to check the status before sending (see function call 3). The character will be sent regardless of whether the transmit character buffer is empty or CTS is high. See function calls 01H and 03H.

Write Directly to the Selected Channel Uart Registers

AH = 10H through AH = 16H on entry to Write Uart Register Directly. Use EXTREME caution if you use this function call.

AL = nnH on entry contains the data to write to the specified register. AH defines the register to write to as 10H=xF8H, 11H=xF9H, etc., where x = 2 or 3 (3F8H, 2F9H). See Appendix A on functions of the Uart registers.

Read Directly from the Selected Channel Uart Registers

AH = 17H on entry to Read Uart Registers Directly.

AL = nnH on entry is the register number to read where 00H=xF8H, 01H=xF9H up to 07H=xFFH. Data is returned from the register read in AL.

Set the Default Selected Channel

AH = 18h Set the default selected channel. This is the channel which will be used anytime that an int 14h function call is made with dx=0 or 1, i.e. dh=0. This provides a way to use any channel with existing software. The SS utility uses this function call to set the active channel if RES14 is active. The default is channel 0. Enter with dx=desired default port. Example DX=0301h is default to channel 3 on com2 card. A subsequent int 14 with dx=1 would access channel 3, i.e. a standard COM2 request would be thru channel 3.

Global Poll

AH = 19h Global poll. Returns the channel number in AX of the highest priority channel whose queue contains data. Upon return, the AL register contains the COM channel (0 or 1) and AH contains the channel number, 0–7). Subsequent servicing of that channel may be accomplished by moving AX into DX and making the desired function call. AX=0FFFFh is returned if no channels have data. On entry, channels are

checked from the starting channel indicated by the DX register through DX=0701h. DX=0000 is first, DX=0100 is checked second DX=0601h, and DX=0701h is checked last. Thus if DX=0 upon entry, 16 channels are checked. If DX = 0001 upon entry, just the 8 channels on com2 are checked.

Return Default or Currently Selected Channel

AH = 1Ah Returns the default or currently selected channel on the board. Enter with dx=0 or 1 for com1 or com2 Returns AL=0–7

AH = 1Bh through 1Fh reserved for future releases.

Read Selected Channel Uart Directly

AH = 28 through AH = 27h Read uart register directly. AL = don't care on entry. 20=xF8, 21=xF9, ... 26=xFE, 27=xFF (the scratch register).

14– Interfacing Notes

General Examples

You can use two PCSS–8's in an interrupt driven mode so that up to 16 ports can be used simultaneously with RES14.COM. Put the command RES14 in your autoexec.bat file. Res14 adds many new function calls to BIOS INT14H.

Access to INT14h from Quick Basic Versions 2.0 and 3.0 (4.0 interface is different) is gained thru USR.OBJ, which must be linked to your Quick Basic program. USR.OBJ contains a function which can be called from your Quick Basic program. Within your basic program you "Call res(ax,dx)" where ax and dx are integer variables. They are passed to and from BIOS INT 14h in the AX and DX registers.

Simple Basic Example:

```
A$="A"
TRANSMIT_FUNC = 1 * 256       'AH=1 TO XMIT
AX% = TRANSMIT_FUNC + ASC(A$) 'load AH with 1, AL with
                  'the ascii code for 'A'
CHANNEL = 5 * 256         'DH gets the channel #
XCOM2=1            'DL gets COM #(0 or 1)
DX% = CHANNEL + XCOM2     'channel 5 on COM2
CALL RES(AX%,DX%)         'Transmit the character 'A'
```

See the QUICK BASIC example program EXAMPLE.BAS for more information. The following is an example of what you can do with int 14 in machine code.

In this case we wanted to run at a baud rate not available using function call 0, so we programmed the baud rate divisor registers directly.

```
setup:
    mov dx,0501h    ;channel 5, com2
    mov al,00101010b   ;9600 baud, even parity ,1s, 7b
    mov ah,0       ;function 0 - initialize (IBM style)
    int 14h     ;Next modify baud rate
    mov ah,17h     ;read register
    mov al,3       ;for xFBh to read LCR
    int 14h
```

```
    or  al,80h      ;set dlab
    mov ah,13h      ;for xFBh to write LCR
    int 14h     ;now dlab is high
    mov al,1        ;115,200 bits per second!
    mov ah,10h      ;for xF8h, write LSB of baud.
    int 14h     ;we know that the MSB of baud
                ;is already 0. now we need to set
                ;DLAB bit low for normal operation
    mov ah,17h      ;read register function
    mov al,3        ;for xFBh, LCR
    int 14h
    and al,7Fh      ;clear dlab
    mov ah,13h      ;write register function for xFBh
    int 14h     ;now dlab is LOW
    mov al,1        ;enable interrupt drive.
    mov ah,5        ;function call 5
    int 14h
    ret
```

USR.OBJ program

```
;*********************************************************
;** COPYRIGHT 1986, 1987, GTEK, INCORPORATED               **
;**     ALL RIGHTS RESERVED, WORLDWIDE                    **
;**         USR.ASM                                       **
;** A program to enable BASIC to interface with the Serial  **
;**Port through the use of the RES14.COM TSR program        **
;**to replace the INT14 IBM Bios driver. See EXAMPLE.BAS for**
;** usage with a machine language program through QuickBasic**
;*********************************************************
(continued on next page)
CODE    SEGMENT PARA PUBLIC 'CODE'
    ASSUME  CS:CODE,DS:CODE

    PUBLIC  RES
RES PROC    FAR
    PUSH    BP      ;save frame reference
    MOV BP,SP       ;this has to be done first
    PUSH    AX      ;save the registers that are used.
    PUSH    BX
    PUSH    DX
    MOV BX,[BP]+6   ;Point to DX% from basic
    MOV DX,[BX] ;Store Index register
    MOV BX,[BP]+8   ;Point to AX% from basic
    MOV AX,[BX] ;Store in AX
    INT 14H     ;Int 14 bios function call
    MOV [BX],AX ;get result from AX to basic AX%
    POP DX      ;dx doesn't change
    POP BX
```

```
    POP AX
    POP BP
    RET 4       ;FIX STACK TO RETURN TO QB
RES ENDP
CODE    ENDS
    END
```

QuickBasic example

```
'* * * * * * * * READ THE FOLLOWING FIRST * * * * * * * * * * *
* *
'You must first  build a library file to use USR from the
'debug mode of QuickBasic. Use BUILDLIB to create a
'USERLIB.EXE FILE that contains USERLIB.OBJ and USR.OBJ
'   C>BUILDLIB USERLIB.OBJ USR.OBJ;
'You can then run QuickBasic to run the example or your own
'program You have to load the user library before you can
'run QB in the debug mode.
'   C>QB EXAMPLE /L USERLIB.EXE
'or you can compile to stand alone EXE files.
'   C>QB EXAMPLE /O;
    C>LINK EXAMPLE USR;
'   C>EXAMPLE

'Study what this program does before you run it.
'I used it on Channel 7 to print to a terminal for testing.
'It initializes all the channels before it actually runs
DEFINT A-Z
TRUE = NOT FALSE
PAUSE = FALSE
GOSUB GTEK
IF AL% = 3 THEN
    PRINT "DRIVER NOT INSTALLED."
    END
END IF
PARMS = &H23   '19200 baud, no parity, 1 stop, 8 data
FUNCT=0
GOSUB ALL.INIT
AX%=PARMS
DX%=&H701      'CHANNEL 7 COM2:
GOSUB INT14.1  'INITIALIZE
GOSUB INT.EN       'ENABLE INTERRUPTS
A$="THE QUICK BROWN FOX JUMPS OVER "
A$=A$+"THE LAZY DOG 0123456789 TIMES. ITERATION #"
FOR J=1 TO 20000
    FOR I=1 TO LEN(A$)
        AL%=ASC(MID$(A$,I,1))
        GOSUB TX.CHAR
    NEXT I
    J$ = STR$(J)
    WHILE LEN(J$)<5
```

```
        J$=" “+J$
    WEND
    J$=J$+CHR$(13)+CHR$(10)
    FOR I=1 TO LEN(J$)
        AL%=ASC(MID$(J$,I,1))
        GOSUB TX.CHAR
    NEXT I
    GOSUB STATUS
    WHILE ((AH% AND 1) = 1)
        GOSUB RX.CHAR
        IF AL% = 19 THEN
            PAUSE = TRUE
        ELSEIF AL% = 17 THEN
            PAUSE = FALSE
        ELSEIF AL% = 3 THEN
            GOSUB INT.DI
            PRINT: PRINT : PRINT
    PRINT ”PROGRAM DE-INITIALIZED AND TERMINATED."
            END
        ELSEIF AL% = 6 THEN
            GOSUB SEND.FILE
        ELSE
            PRINT CHR$(AL%);
        END IF
        GOSUB STATUS
        IF PAUSE THEN AH%=1
    WEND
NEXT J
GOSUB INT.DI
PRINT: PRINT : PRINT
PRINT “PROGRAM DE-INITIALIZED AND TERMINATED.”
END

SEND.FILE:
FILES
LINE INPUT “ENTER FILE NAME :”;IFN$
OPEN “I”,1,IFN$
SF1:
LINE INPUT #1, TX.LINE$
GOSUB TX.LINE
WHILE INKEY$=""
WEND
IF EOF(1) THEN CLOSE: RETURN
GOTO SF1
```

```
ALL.INIT:
AL.SAV% = PARMS
AH.SAV% = FUNCT
FOR I% = 0 TO 7
    AL% = AL.SAV%
    AH% = AH.SAV%
    DH% = I%
    DL% = 1
    GOSUB INT14
    PRINT "PORT NUMBER ";I;" ";HEX$(AX%)
NEXT I%
AH%=3: GOSUB INT14: PRINT HEX$(AX%)
IF (AX% AND 1) = 1 THEN
    AH%=2: GOSUB INT14
ELSE
    PRINT "NO CHAR TO RECEIVE"
END IF
AH%=3: GOSUB INT14: PRINT HEX$(AX%)
RETURN

INIT:
AX% = &H0023: DX% = &H0701
GOTO INT14.1

TX.LINE:
FOR I=1 TO LEN(TX.LINE$)
    AL%=ASC(MID$(TX.LINE$,I,1))
    GOSUB TX.CHAR
NEXT I
AL%=13: GOSUB TX.CHAR
AL%=10

TX.CHAR:
AH% = 1: GOTO INT14

RX.CHAR:
AH% = 2: GOTO INT14

STATUS:
AH% = 3: GOTO INT14

GTEK:
AX% = &H400: GOTO INT14.1

INT.EN:
AX% = &H501: GOTO INT14.1

INT.DI:
AX% = &H500: GOTO INT14.1
```

```
DTR:
AH% = 6: GOTO INT14

TIME:
AH% = 7: GOTO INT14  'AL=TIME IN SECONDS

RTS.ON:
AX% = &H801: GOTO INT14.1

RTS.OF:
AX% = &H800: GOTO INT14.1

BRK.ON:
AX% = &H901: GOTO INT14.1

BRK.OF:
AX% = &H900: GOTO INT14.1

LOOP.BACK.ON:
AX% = &HA01: GOTO INT14.1

LOOP.BACK.OF:
AX% = &HA00: GOTO INT14.1

RQC:
AX% = &HB00: GOTO INT14.1


QFL:
AX% = &HC00: GOTO INT14.1

DTR.ON:
AX% = &HD01: GOTO INT14.1

DTR.OF:
AX% = &HD00: GOTO INT14.1

INT14:
AX% = AH% * 256 + AL%
DX% = DH% * 256 + DL%

INT14.1:
CALL RES(AX%, DX%)
AH% = INT (AX% / 256)
AL% = AX% - (AH% * 256)
RETURN
```

15– Xenix/UNIX

Installation with MicroPort System V Unix.

The following example assumes that you have a COM1: channel in the computer which is set as “t0”, and you wish to install the PCSS–8–U as “t1—t8”.

1.To generate drivers for the board, use command:

ttypatch a704, i3, t1, n8

2.Modify the “inittab” file to contain lines for “t1–t8”. You can duplicate the line for “t0” as it comes from MicroPort. Change the “t0”’s and “tty0”’s to “t1” and “tty1”, “t2” and “tty2”, ... “t8” and “tty8” as shown for example:

t0:23:respawn:/etc/getty tty0 9600
t1:23:respawn:/etc/getty tty1 9600
t2:23:respawn:/etc/getty tty1 9600
...
t8:23:respawn:/etc/getty tty8 9600

3.Run “init q” to activate the ports or “shutdown” and the next time you boot, the ports will become active.

Any additional information needed about “ttypatch” and “inittab” or the answers to any questions should be taken from the MicroPort documentation.

Installation with Concurrent Dos–386

Use a PCSS–8H (or 8TH) and install it as a Hostess–8 card. See appendix H.

Installing PCSS–8TH/8H under SCO–Xenix

1—To install the PCSS–8TH (and/or a PCSS–8H) as a COM1 or COM2 board under SCO–XENIX, first set the correct jumper configuration as above.

2—After jumper configuration, install into your computer. Remember you can't have another COM card on the same COM port. Disable all other COM cards or system board for the COM port you are installing.

3—Boot the computer and log in as the super user in the single user mode. Enter the command "/etc/mkdev serial" to enable the ports on the PCSS–8TH. Select Option 5 to install an 8 port board. Next select either COM1 or COM2. Re–boot in the multi–user mode to use the installed ports.

1

A

B

C

D

I

J

L

M

P

R

S

T

U

W

X