Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.


This document contains proprietary information, which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

Portable Computer Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.

This addendum describes four things:

- Problems in the HP-94 built-in software (operating system and BASIC interpreter version V1.03).
- Problems in the HP-94 Software Development System software (HNWN version 1.00).

This addendum discusses problems from the perspective of assembly language programs. Problems that affect BASIC language programs are described in the Software Development System addendum from the perspective of BASIC programs.

### How to Determine the Software or Document Version Number

To see if these problems affect your software development on the HP-94, you must verify the version number of the software or document you are using. Here is how to determine the version number:

- For the HP-94 built-in software, turn the machine off. Then hold down the **CLEAR** and **ENTER** keys and turn it on. The following message should appear in the display:

  \[
  \text{Copr. HP 1985 Vx.xx}
  \]

  The \( \text{Vx.xx} \) is the built-in software version number. This addendum discusses problems in V1.03.

- For the Software Development System software, use the MS-DOS **DIR** command to look at the date and time shown for the files on the Software Development System (HP 82520A) or Bar Code Software (HP 82520S) master discs. This addendum discusses problems in HNWN .EXE with a date of 12-08-86 and a time of 3:13p.

- For the Technical Reference Manual, the document edition number and date are on the first page of the manual. This addendum discusses errors in Edition 1, February 1987.

### Contents of Patch Utilities Disc

This addendum includes a disc with six files, as follows:

- SYPA.EXE — the patch keyword for three of the operating system problems. The addendum describes how to use SYPA, which only performs its operations for version V1.03 of the operating system.

- HNSG.EXE — the enhanced serial port handler. The addendum includes chapters from the HP-94 Datacomm Utilities Pac manual that describes how to use HNSG.
- **SYSG.EXE** — the keyword for configuring HNSG. The addendum includes chapters from the *HP-94 Datacomm Utilities Pac* manual that describes how to use SYSG.

- **SYPA.MDS** — RAM MDS file containing SYPA.

- **HNSG.MDS** — RAM MDS file containing HNSG.

- **SYSG.MDS** — RAM MDS file containing SYSG.

The MDS files can be placed in HP-94 memory unchanged using the HXCOPY program on the development system computer and the C (copy) operating system command on the HP-94. The EXE files must be processed by HXC before being sent to the HP-94. SYPA and SYSG should be specified as type A, and HNSG should be specified as type H.

---

**Right-to-Reproduce Software License**

The following copyright notice and software license agreement applies to the software contained on the *Patch Utilities* disc:

"(c) Copyright Hewlett-Packard Company, 1986, 1987. All rights are reserved. Copying or other reproduction of this program for inclusion in an application or for archival purposes is permitted without the prior written consent of Hewlett-Packard Company."
Operating System Problems

There are 14 operating system problems discussed here. Seven relate to the built-in serial port handler, two to releasing unused memory, one to the beeper, one to user-defined characters, two to the resident debugger, and one to a reserved interrupt.

Serial Port Write After Error 218 Locks Up HP-94

If the WRITE function (13h) reports error 218 (lost connection while transmitting) while writing to channel 1 using the built-in serial port handler, subsequent writes to channel 1 will lock up the machine until the system timeout expires. For example, the following routine is intended to wait for the host to be available before transmitting a file by sending a \texttt{C} out the serial port to the host. The expected behavior is that the program will execute the wait loop until the host is available:

```
        push cs
        pop es
        mov bx, offset NULL ;ES:BX = null string
        push cs
        pop ds
        mov dx, offset PARAMS ;DS:DX = open parameters
        mov ah, 0Fh ;OPEN function code
        mov al, 01h ;Channel 1 = serial port
        int 1Ah ;Open the channel
        or al, al ;Any errors?
        jnz OPENERR ;Yes
        mov bx, offset CRLF ;ES:BX = CRLF
        DOWRITE:
        mov ah, 13h ;WRITE function code
        mov al, 01h ;Channel 1 = serial port
        mov cx, 02h ;CRLF is two bytes long
        int 1Ah ;Write the CRLF
        cmp al, 218 ;Error 218?
        je DOWRITE ;Yes, so try again
        ....

PARAMS db 1 ;9600 baud
        db 00001101b ;XON/XOFF, 7 bits, even parity
            ;1 stop bit, null strip disabled
NULL db 00h ;No terminate character
CRLF db 0Dh, 0Ah ;CRLF
```

If the host is not ready, its clear to send (CTS) control line will not have been raised, and error 218 will be reported because the \texttt{C} cannot be transmitted. When the program tries to send the \texttt{C} a second time, the program will not return from the WRITE function until the current system timeout expires (default 2 minutes). When the timeout expires, either the machine will turn off, or the power switch/timeout interrupt routine will be executed if one was defined using the SET_INTR function (0Ah), in which case the WRITE function will return with error 76h (118).
This behavior is dependent upon the number of writes to the serial port, not the number of characters sent by each write. On the first write, error 218 occurs on the first character of the string sent to the serial port. Because of the error, the remainder of the string will not be sent, so the HP-94 does not lock up. On the second write, the HP-94 does lock up, but only on the first character of the string sent.

**Preferred Solution**

Use HNSG, the enhanced serial port handler supplied with the *HP-94 Datacomm Utilities Pac*, instead of the built-in serial port handler. HNSG will not lock up the HP-94 in this situation.

**Alternate Solution**

Call the internal entry point of the keyword SYPA, supplied on the disc that accompanies this addendum. This code will reset the 82C51A so it can be written to again after error 218 occurs. This reset will only occur for the built-in serial port handler, and only if the handler is open. The keyword will not affect the current baud rate, software handshaking, data format (number of data bits, stop bits, and parity), null stripping, terminate character, or the control lines being used.

SYPA will lower the request to send (RTS) and data terminal ready (DTR) control lines briefly (for ~20 μs). It will also erase the byte that is unable to be transmitted (the one that is "stuck" in the 82C51A). It is still the responsibility of the application program to send that byte once the communication link is established (i.e., when CTS goes high from the host computer). If the 82C51A is in the process of receiving a byte when SYPA is executed, that byte will be lost. SYPA should not be used except when error 218 occurs.

To use SYPA for the above example, modify its source code as follows:

```asm
cmp at, 218 ; Error 218?
jne NOT218 ; No
push cs ; SS:SP = segment address of SYPA name
mov ax, offset SYPA_NAME
push ax ; SS:SP = offset address of SYPA name
push ax ; SS:SP+2 = segment address of SYPA name
call INTERNAL ; Call SYPA internal entry point
; to reset the 82C51A
jmp DOWRITE ; Go send the CRLF again

NOT218:
.
.
.
SYPA_NAME db "SYPA" ; SYPA file name
include internal.asm ; Include the INTERNAL utility from
; the Technical Reference Manual disc
```

4 Technical Reference Manual Addendum
Error 218 When Sending XON or XOFF Locks Up HP-94

If error 218 (lost connection while transmitting) occurs because the host shut down its communications just as the built-in serial port handler sent two consecutive handshake characters, the HP-94 will take one of two actions:

- If the handshake characters are two XOFFs, sent by the serial port handler interrupt service routine, the HP-94 will lock up, requiring that the reset switch be pressed. No timeout, power switch, or low battery event, default or user-defined, will cause it to turn off (except the automatic turn off that occurs 2-5 minutes after low battery).

- If the handshake characters are an XOFF followed by an XON (the XOFF sent by the interrupt service routine, and the XON sent when the READ function (12h) empties the receive buffer), the HP-94 will lock up and not turn off. The default power switch or low battery events will still behave normally. No timeout event, default or user-defined, will be executed. The user-defined power switch or low battery interrupt routines will be executed, but the READ function will not end and return error 77h (119) or C8h (200). Depending on how the application has defined power switch or low battery behavior, this problem may require that the reset switch be pressed.

Preferred Solution

Use HNSG, the enhanced serial port handler supplied with the HP-94 Datacomm Utilities Pac, instead of the built-in serial port handler. HNSG will not lock up the HP-94 in this situation.

Alternate Solution

The host should not shut down the communication session before receiving an XON if it has received an XOFF. If the HP-94 shuts down the session without sending an XON, then the host need not wait for the XON (see the "XON Not Sent to Host When Serial Port Closed" problem description).

XON Not Sent to Host When Serial Port Closed

If the built-in serial port handler has sent an XOFF to the host, and then the handler is closed with the CLOSE function (10h), the handler will not send an XON to the host. This may cause the host to lock up because it expects an XON to reenable transmission. (Note: if the host sent an XOFF, the built-in serial port handler also does not wait for an XON when it is closed.)

Preferred Solution

Use HNSG, the enhanced serial port handler supplied with the HP-94 Datacomm Utilities Pac, instead of the built-in serial port handler. The currently available version of HNSG (version A.02.00) does not send an XON when it is closed, but version A.03.00 will (available fourth quarter 1987).
Alternate Solution

Just before closing the serial port handler, send an XON to the host. For example, use the WRITE function (13h) to send an XON (11h) to the serial port, then use the CLOSE function to close the handler:

```
push cs
pop es
mov bx,offset XON ;ES:BX = XON
mov ah,13h ;WRITE function code
mov al,01h ;Channel 1 = serial port
mov cx,01h ;XON is one byte long
int 1Ah ;Write the XON, ignoring errors
mov ah, 10h ;CLOSE function code
mov al,01h ;Channel 1 = serial port
int 1Ah ;Close channel 1
```

It is only necessary to do this if the host will only end a datacomm session normally after receiving an XON to match a received XOFF.

Alternatively, the host should not expect to receive an XON if it has received an XOFF. However, it must maintain the communication session to receive an XON if one is sent by the HP-94 to prevent the HP-94 from locking up (see the "Error 218 When Sending XON or XOFF Locks Up HP-94" problem description).

Datacomm Starts Unexpectedly at Turn-On

When the HP-94 is turned on, the serial port hardware is enabled for a short period of time, then disabled. The request to send (RTS) and data terminal ready (DTR) control lines may or may not be raised, depending on the state of the serial port hardware at that time. If the lines are raised, they will remain raised even after the port is disabled. If the host will take some action when it sees these control lines, it may start or complete a datacomm session before the HP-94 is actually ready.

For example, the normal procedure to copy an MDS file into the HP-94 from a development system computer is to enter the C (copy) command on the HP-94, then enter the HXCOPY command on the development system. Suppose the HP-94 is turned on in command mode (hold [CLEAR] and [ENTER] while turning it on), and control lines are actually raised. If HXCOPY is started before entering the C command, the development system computer will assume the HP-94 is ready to receive the MDS file, and send it to the HP-94 anyway (although the file will be discarded since the HP-94 is not ready to receive it). The raised state of the control lines has indicated to the serial port on the development system computer that the transmission can proceed.

6 Technical Reference Manual Addendum
This only occurs when the HP-94 is turned on and the control lines are actually raised. The control lines will remain raised until the port is opened, then closed again, at which time the lines will be lowered. Whether or not this is a problem depends on the behavior of the serial port on the host.

**Solution**

For the above example, this problem is avoided by always executing the C command first, then the HXCOPY command (i.e., start the receiving machine, then the sending machine).

In general, if the host serial port will be affected by this condition, the control lines must be lowered before the HP-94 serial port is physically connected to the host serial port. This can be done by opening the serial port using the OPEN function (0Fh), then closing it again using the CLOSE function (10h). This will cause RTS and DTR to be lowered; if they are already low, this procedure will raise them briefly, then lower them again. Consequently, if this is done after connecting to the host, then the same false readiness indication will be given to the host as is given at power on.

---

**Serial Port Errors Ignored Until Input Operation Starts**

If serial port errors 201-208 (parity, framing, and overrun errors) occur after opening the built-in serial port handler, but before executing the READ function (12h), the errors will be ignored. Errors that occur after READ starts are reported, although possibly for the wrong byte of data (see the "Serial Port Errors Reported for Wrong Byte of Data" problem description).

**Solution**

Use HNSG, the enhanced serial port handler supplied with the *HP-94 Datacomm Utilities Pac*, instead of the built-in serial port handler. HNSG will report serial port errors properly in this situation.

---

**Serial Port Errors Reported for Wrong Byte of Data**

If serial port errors 201-208 (parity, framing, and overrun errors) occur while the READ function (12h) is waiting for data to be received, they may be reported for the wrong byte of data. For example, suppose 39 bytes of data have been received in the receive buffer, and READ requests 20 bytes. After 5 bytes have been transferred from the receive buffer into the read buffer specified by the READ caller, byte 40 arrives in the receive buffer with a parity error. When byte 6 is transferred from the receive buffer into the read buffer, byte 40's parity error is reported. READ returns 5 bytes and a parity error. The error is on the wrong byte, and does not even apply to the 20 bytes requested.
Preferred Solution

Use HNSG, the enhanced serial port handler supplied with the HP-94 Datacomm Utilities Pac, instead of the built-in serial port handler. HNSG will report serial port errors properly in this situation.

Alternate Solution

Read data from the receive buffer one character at a time. For example, use the READ function, but specify one byte as the read buffer length. This will ensure that the error is reported for the proper byte, as long as the data arrives as a slower rate than it is read from the receive buffer. If this problem occurs because the data arrives faster than it can be read, a lower baud rate could also be used to avoid the problem.

Characters Discarded by Serial Port When Display Cleared

When the home-and-clear control character (0Ch) is sent to the display, as many as 45 characters sent to the serial port by the host may be discarded by the HP-94 (not received). This is because interrupts are disabled for ~45 ms while the control character is processed.

Solution

Use the following sequence to clear the display:

```
  sub  cl, cl  ; Column 0
  mov  ch, 03h ; Row 3
  mov  ah, 05h ; CURSOR function code
  mov  al, 01h ; Move cursor
  push  ax ; Save function information
  int  1Ah ; Move cursor to (0, 3)
  push cs
  pop es
  mov bx, offset FOURLF ; ES:BX = 4 line feeds
  mov ah, 04h ; PUT_LINE function code
  int  1Ah ; Write 4 line feeds
  sub cx, cx ; Column 0, row 0
  pop ax ; CURSOR function information
  int  1Ah ; Move cursor to (0, 0)
```

FOURLF db 0Ah, 0Ah, 0Ah, 0Ah, 00h
Releasing Unused Memory Locks Up HP-94

When the operating system attempts to release scratch areas, it sometimes alters the operating system scratch space, causing the HP-94 to lock up. This may occur when the scratch areas cannot be returned to free space, but instead are flagged as free blocks. Scratch areas are released under the following conditions:

- Closing a user-defined handler with the CLOSE function (10h) or the &CLOSE command of the SYSC script processing utility from the HP-94 Datacomm Utilities Pac.
- Deleting files with the DELETE function (14h) or the E (erase) operating system command.
- Releasing scratch areas with the REL_MEM function (0Ch).

Refer to "Scratch Areas" in the "Memory Management" chapter of the Technical Reference Manual for details on scratch areas and free blocks.

Solution

Call the internal entry point of the keyword SYPA, supplied on the disc that accompanies this addendum. This routine will clear an area of memory so that the process of releasing scratch areas will end without altering the operating system scratch space.

To use SYPA, add the following lines once at the beginning of the application:

```
push cs
mov ax, offset SYPA_NAME
push ax
    ;SS:SP = segment address of SYPA name
    ;SS:SP = offset address of SYPA name
    ;SS:SP+2 = segment address of SYPA name
    ;Call SYPA internal entry point
    ;to clear the memory area
    call INTERNAL
    SYPA_NAME db SYPA
    ;SYPA file name
include internal.asm
    ;Include the INTERNAL utility from
    ;the Technical Reference Manual disc
```

This should be added in the application before it executes the OPEN (0Fh) or GET_MEM (0Bh) functions, or before it calls the internal entry point of any keyword that allocates scratch areas (such as SYBC, SYSC, SYSG, SYSP, or SYWN). If the program turns the HP-94 off and specifies a subsequent warm start, this line should also be added to the routine that turns the HP-94 off, after the END_PROGRAM function call.

When closing a user-defined handler, SYPA by itself may not be sufficient to prevent this problem if the handler scratch areas cannot be returned to free space. For example, the problem may occur if a program opens a user-defined handler, allocates a scratch area with GET_MEM, then closes the handler, thereby trapping the handler scratch areas and causing them to be flagged as free blocks. With SYSC, the problem may occur if the handler is opened in a script file, but closed in the program after SYSC ends, or if the handler is opened in the program, then closed in the script file.

As a precaution, close a user-defined handler in the same routine where it was opened. If it was
opened in the main program, close it there. If it was opened in a subprogram, close it there before ending the subprogram. If it was opened in a script file, close it there before ending the script. As another precaution, release scratch areas in the reverse order they were created (on a last-in, first-out basis), taking into account the fact that opening and closing user-defined handlers will also cause scratch areas to be allocated and released.

When deleting files using the DELETE function, using SYPA is sufficient to prevent this problem from occurring. Before deleting files using the E command, SYPA can be executed from command mode by typing SSYPA [ENTER].

NOTE
When closing user-defined handlers or releasing scratch areas, the likelihood of this problem occurring is extremely low, even if SYPA and the suggested precautions are not used. If closing handlers or releasing scratch areas does not cause this problem to appear when testing the application, then it will not cause the problem when the application is used.

When deleting files, the likelihood of this problem occurring is even lower than for closing handlers or releasing scratch areas, even if SYPA is not used, although its occurrence cannot be verified thoroughly during application testing because it is impossible to test all possible data file combinations.

Deleting Data Files Causes Memory to Vanish
When data files are deleted by running programs, the operating system scratch space may be altered erroneously. This will result in a small portion of memory being unavailable for use until the next cold start.

Solution
Follow the precautions described for closing user-defined handlers and releasing scratch areas in the "Releasing Unused Memory Locks Up HP-94" problem description.

NOTE
When deleting data files, the likelihood of this problem occurring is extremely low, even if the suggested precautions are not used.

Beeper Not Turned Off When Entering Command Mode
When the beeper is started by a program that ends and returns to command mode before the beeper duration expires, the beeper will continue for the specified duration.
Solution
Do not start the beeper for long durations just prior to ending a program.

User-Defined Characters Not Available at Warm Start
When the HP-94 has user-defined characters in file SYFT, and is turned off using the END_PROGRAM function (00h) to specify a subsequent warm start, the display software will not use the user-defined characters when the HP-94 is turned back on. The display software will behave as if there are no user-defined characters — blanks will be displayed for those ASCII codes.

Solution
Call the internal entry point of the keyword SYPA, supplied on the disc that accompanies this addendum. This routine will locate the SYFT file and, if found, inform the display software of the presence of user-defined characters.

To use SYPA, add the following lines after the END_PROGRAM function call that turns the HP-94 off and specifies a subsequent warm start:

```asm
push cs          ;SS:SP = segment address of SYPA name
mov ax,offset SYPA_NAME
push ax          ;SS:SP = offset address of SYPA name
                        ;SS:SP+2 = segment address of SYPA name
call INTERNAL    ;Call SYPA internal entry point
                        ;to activate the user-defined font

SYPA_NAME db USYPAH ;SYPA file name
include internal.asm ;Include the INTERNAL utility from
                        ;the Technical Reference Manual disc
```

Debugger Changes Bar Code Port Status
When the X command in the resident debugger is used to change the debugger console from the HP-94 keyboard/display to the serial port, it may enable bar code port power, bar code transition detection, or both.

Solution
If it is important to prevent this condition while debugging, the application should use the following routine, which will place the saved copy of the main control register into another part of the operating system scratch space for the debugger to use.

Technical Reference Manual Addendum 11
Debugger Turns Off Serial and Bar Code Ports

When the X command in the resident debugger is used to change the debugger console from the serial port to the HP-94 keyboard/display, it will turn off the serial and bar code port power and bar code transition detection.

Solution

If the serial port is used as the debugger console, do not restore the HP-94 keyboard/display as console while the program being debugged is running.

Default INT 56h Interrupt Routine Clears Low Battery Interrupt

The default interrupt service routine for interrupt 56h, the first reserved interrupt, clears both interrupt 56h and interrupt 54h, the low battery interrupt.

Solution

The HP-94 does not currently use interrupt 56h — it is reserved for future use. If a hardware device is developed that uses interrupt 56h, the program that controls the device should not enable the interrupt using the default interrupt service routine. Instead, take over the interrupt vector, and use an interrupt service routine in the program that only clears interrupt 56h.
BASIC Interpreter Problems

There are two BASIC interpreter problems discussed here. One is a condition that should be tested for when using the BASIC interpreter utility routines, and one can be avoided by using an operating system function.

Integer Value -32768 Not Negated Properly

When the SNEG BASIC interpreter utility routine attempts to negate the integer value -32768, it will return -32768, not +32768.

Solution

If negative integers approaching -32768 are used in the application, check for that value explicitly before calling SNEG.

No Beep for Non-Numeric Errors

The ERROR BASIC interpreter utility routine does not beep, even though the DISPLAY_ERROR operating system function (18h) does beep.

Solution

Just before calling ERROR, write the low beep display control code (07h) to the display.

```assembly
mov ah,03h       ;PUT_CHAR function code
mov al,07h       ;Low tone
int 1Ah          ;Call PUT_CHAR function
```
Software Development System Problems

There is one Software Development System problem discussed here. It can be avoided by not using the utility in certain ways.

HNBC and HNSP IOCTL Not Accessible Through HNWN

If a program calls the IOCTL routine in either of the low-level bar code handlers HNBC or HNSP while HNWN is being used as a high-level handler, HNWN will not return the results of the IOCTL routine. Instead, HNWN will return incorrect results.

(The IOCTL routine in a low-level handler can be called using the XIOCTL utility routine provided on the Technical Reference Manual disc.)

Solution

If the program needs to call the IOCTL routine in HNBC or HNSP, it should not open the low-level handler with HNWN as the high-level handler. If this is not possible, below are workarounds to use for the different IOCTL functions provided by HNBC and HNSP. Note that closing the high- and low-level handler pair will turn off the Smart Wand attached to the bar code or serial port, which will cause the Smart Wand to be reset to its power-on configuration, and the low-level handler to flush its receive buffer.

<table>
<thead>
<tr>
<th>IOCTL Function</th>
<th>Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE_CONFIG (02h)</td>
<td>Close high- and low-level handler pair&lt;br&gt;Use SYBC or SYSP to change low-level handler configuration&lt;br&gt;Open high- and low-level handler pair</td>
</tr>
<tr>
<td>GET_CONFIG (01h)</td>
<td>Close high- and low-level handler pair&lt;br&gt;Open low-level handler&lt;br&gt;Use XIOCTL utility to call GET_CONFIG function&lt;br&gt;Close low-level handler&lt;br&gt;Open high- and low-level handler pair</td>
</tr>
<tr>
<td>IDENTIFY (00h)</td>
<td>Close high- and low-level handler pair&lt;br&gt;Open low-level handler&lt;br&gt;Use XIOCTL utility to call IDENTIFY function&lt;br&gt;Close low-level handler&lt;br&gt;Open high- and low-level handler pair</td>
</tr>
<tr>
<td>RECEIVE_FLUSH (04h)</td>
<td>Close high- and low-level handler pair&lt;br&gt;Open high- and low-level handler pair</td>
</tr>
<tr>
<td>RECEIVE_STATUS (03h)</td>
<td>Do not use HNWN with the low-level handler</td>
</tr>
<tr>
<td>WR_RD_EN (80h) (HNSP only)</td>
<td>Close high- and low-level handler pair&lt;br&gt;Open low-level handler&lt;br&gt;Use XIOCTL utility to call WR_RD_EN function&lt;br&gt;Close low-level handler&lt;br&gt;Open high- and low-level handler pair</td>
</tr>
</tbody>
</table>
Technical Reference Manual Errors

There are ten Technical Reference Manual errors discussed here. Three are in example programs, six are clarifications, and two are in a utility routine.

Operating System, Chapter 3 — User-Defined Handlers

3-14: "Cautions"
Add this paragraph at the end of the caution: "When returning to command mode, interrupts will be disabled (CLI) when the CLOSE routines of open handlers are called. If the CLOSE routine needs interrupts enabled, it must explicitly do so (with STI)."

Operating System, Chapter 4 — Operating System Functions

4-16: FIND_NEXT function example, version number in program header
"db 0100h" should be "dw 0100h".
4-28: OPEN function example, OPEN function code equate
The equate should be 0Fh, not 0Eh.
4-32: READ function, registers returned, AL = 76h (118)
This error code should have a dagger footnote mark (†) next to it.

Operating System, Chapter 10 — Serial Port

10-1: "Initializing the Serial Port", last item on page
Because of the way the 82C51A is initialized when the HP-94 is turned on, the initial state of the 82C51A is the command instruction state, not the mode instruction state. To reset it from this state, write a 40h to the serial port control register (11h). Then send the proper mode instruction byte to configure its operating behavior (data bits, parity, etc.). For details, refer to the OKI MSM82C51A data sheet in the "Hardware Specifications" in the Technical Reference Manual.

To guarantee proper initialization of the 82C51A regardless of what state it might have been left in by another handler, write three consecutive nulls (00h) to the serial port control register. This will always put the 82C51A into command instruction state. Then send the reset command (40h), followed by the desired mode instruction.

BASIC Interpreter, Chapter 3 — Assembly Language Subprograms

3-5: "Access to BASIC Interpreter Utility Routines"
It is implied that a program use a FAR CALL to call a BASIC interpreter utility routine. However, HXC will reject a FAR CALL, because that instruction must be relocated at run time. The calling program should use an indirect FAR CALL, which does not require relocation. For
example, the following code will call the GETARG utility routine:

```
sub cl, cl           ; Want positive word from GETARG
mov di, sp          ; DI = stack pointer
mov ax, ss: [di+02h] ; Get CSEG
push ax             ; Put on stack
sub sp, 02h         ; Make room for GETARG offset
mov word ptr ss: [di-04h], 3Ch; GETARG offset
call dword ptr ss: [di-04h]; Indirect FAR CALL to CSEG:3Ch
```

### Appendix L — Hewlett-Packard Bar Code Handlers

L-17: "Response to Escape Sequences From Smart Wand", "No Read Message"
The default no-read message is the null string, and is preceded by the header and followed by the trailer. Since the default header is the null string, and the default trailer is ®†, the effective default no-read message is R 'r.

L-18: Table L-14
Add "Configuration Dump, 4, High" to the table.

### Appendix M — Disc-Based Utility Routines

M-6: "HP-94 ADDRESSES", equates that refer to OSRAM_SEG
The comments for RESTART_STATUS, MAXDIR, EVENT_TBL, and OPENTBL should refer to OSSCRATCH_SEG, not OSRAM_SEG.

M-6: "HP-94 ADDRESSES", the last four equates
The TIMEOUT symbol, shown as 3Ah, conflicts with the TIMEOUT function code equate on page M-7. These four offsets within the system ROM have been renamed as follows:

<table>
<thead>
<tr>
<th>Old Name</th>
<th>New Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY_SCAN</td>
<td>KEY_SCAN_JMP</td>
</tr>
<tr>
<td>CURSOR_BLINK</td>
<td>CURSOR_BLINK_JMP</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>TIMEOUT_JMP</td>
</tr>
<tr>
<td>VERSION</td>
<td>ROM_VERSION</td>
</tr>
</tbody>
</table>

The new names appear in EQUATES.ASM on the disc supplied with the Technical Reference Manual — only the listing in appendix M is in error.
Additional Technical Reference Manual Information

This section presents additional information that was not available in Edition 1 of the Technical Reference Manual. The first is an update to the handler resource usage of Hewlett-Packard handlers, and the second is a modification to one of the utilities.

Handler Resource Usage

Here is an updated version of the handler resource usage shown in appendix K (table K-2) that reflects information about version A.03.00 of HNSG (available fourth quarter 1987), and its accompanying high-level handler HNCC.

Hewlett-Packard Handler Resource Usage

<table>
<thead>
<tr>
<th>Handler Name</th>
<th>Handler Identifier</th>
<th>Valid Data Flag</th>
<th>IOCTL Function Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>HNBC</td>
<td>BC</td>
<td>FFh</td>
<td>00h-04h</td>
</tr>
<tr>
<td>HNSP</td>
<td>SP</td>
<td>FFh</td>
<td>00h-04h,80h</td>
</tr>
<tr>
<td>HNSG (A.02.00)</td>
<td>SG</td>
<td>FEh</td>
<td>00h-04h,81h</td>
</tr>
<tr>
<td>HNSG (A.03.00)</td>
<td>SG</td>
<td>FEh</td>
<td>00h-04h,81h-84h</td>
</tr>
<tr>
<td>HNCC</td>
<td>CC</td>
<td>FDh</td>
<td>40h-44h</td>
</tr>
<tr>
<td>Reserved</td>
<td>—</td>
<td>80h-FCh</td>
<td>07h-3Fh,47h-7Fh</td>
</tr>
</tbody>
</table>

Up to now, function codes 00h-06h have been reserved for standard IOCTL functions for low-level handlers. This range has now been expanded to 00h-3Fh to allow extra general-purpose IOCTL functions to be defined for low-level handlers. Function codes 40h-7Fh are also now reserved for the equivalent functions for high-level handlers. The range 80h-FFh is for handler-specific IOCTL functions. This usage is summarized below:
Reserved IOCTL Function Codes

<table>
<thead>
<tr>
<th>Function</th>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>IDENTIFY</td>
<td>Low-level handler</td>
</tr>
<tr>
<td>01h</td>
<td>GET_CONFIG</td>
<td>Low-level handler</td>
</tr>
<tr>
<td>02h</td>
<td>CHANGE_CONFIG</td>
<td>Low-level handler</td>
</tr>
<tr>
<td>03h</td>
<td>RECEIVE_STATUS</td>
<td>Low-level handler</td>
</tr>
<tr>
<td>04h</td>
<td>RECEIVE_FLUSH</td>
<td>Low-level handler</td>
</tr>
<tr>
<td>05h</td>
<td>SEND_STATUS</td>
<td>Low-level handler</td>
</tr>
<tr>
<td>06h</td>
<td>SEND_FLUSH</td>
<td>Low-level handler</td>
</tr>
<tr>
<td>07h-3Fh</td>
<td>Spare</td>
<td>Low-level handler</td>
</tr>
<tr>
<td>40h</td>
<td>IDENTIFY</td>
<td>High-level handler</td>
</tr>
<tr>
<td>41h</td>
<td>GET_CONFIG</td>
<td>High-level handler</td>
</tr>
<tr>
<td>42h</td>
<td>CHANGE_CONFIG</td>
<td>High-level handler</td>
</tr>
<tr>
<td>43h</td>
<td>RECEIVE_STATUS</td>
<td>High-level handler</td>
</tr>
<tr>
<td>44h</td>
<td>RECEIVE_FLUSH</td>
<td>High-level handler</td>
</tr>
<tr>
<td>45h</td>
<td>SEND_STATUS</td>
<td>High-level handler</td>
</tr>
<tr>
<td>46h</td>
<td>SEND_FLUSH</td>
<td>High-level handler</td>
</tr>
<tr>
<td>47h-7Fh</td>
<td>Spare</td>
<td>High-level handler</td>
</tr>
<tr>
<td>80h-FFh</td>
<td>(handler-specific)</td>
<td>Low- or high-level handler</td>
</tr>
</tbody>
</table>

---

**HP-94 Console Status**

A small modification of the SCANKYBD utility provided on the *Technical Reference Manual* disc will allow determining six aspects of HP-94 console status:

- Display backlight status — on or off (changed by writing 1Eh or 1Fh to the display)
- Keyboard shift status — unshifted or shifted (changed by writing 0Eh or 0Fh to the display)
- Display software status — busy or not busy (interrupt service routines should not write directly to the LCD controllers or call display software routines while the built-in display software is busy updating the display)
- Cursor blink status — cursor currently displayed or not (will always indicate not displayed if cursor blink is disabled)
- Cursor blink control — cursor blink disabled or not (changed by writing 01h or 02h to the display)
- Key repeat status — key has repeated or not

To modify SCANKYBD.ASM, add the following two lines immediately after the call dword ptr ss:[si]:

```
mov si, es:[1Ch]   ; Get offset of console status
mov ah, ds:[si]   ; Get console status byte
```

With this modification, the console status byte will be returned in AH. The meanings of the bits in the console status byte are shown below:

---

18 Technical Reference Manual Addendum
Console Status Byte Returned by Modified SCANKYBD Utility

X = ignore
HNSG: An Enhanced Serial Port Handler
## Contents

**Chapter 5**

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-01</td>
<td>Features of HNSG</td>
</tr>
<tr>
<td>5-02</td>
<td>Installing HNSG</td>
</tr>
<tr>
<td>5-05</td>
<td>HNSG Parameter Bytes</td>
</tr>
<tr>
<td>5-06</td>
<td>Accessing HNSG from BASIC</td>
</tr>
<tr>
<td>5-07</td>
<td>The SYSG Utility</td>
</tr>
<tr>
<td>5-12</td>
<td>Setting OPEN Parameters with SYSG</td>
</tr>
<tr>
<td>5-13</td>
<td>Changing Parameters with SYSG</td>
</tr>
<tr>
<td>5-14</td>
<td>Reading Current Parameters with SYSG</td>
</tr>
<tr>
<td>5-15</td>
<td>Sending a BREAK with SYSG</td>
</tr>
<tr>
<td>5-16</td>
<td>Reading the Buffer Status with SYSG</td>
</tr>
<tr>
<td>5-17</td>
<td>Flushing the Input Buffer with SYSG</td>
</tr>
<tr>
<td>5-17</td>
<td>Opening HNSG</td>
</tr>
<tr>
<td>5-17</td>
<td>Accessing HNSG From Assembly Language</td>
</tr>
<tr>
<td>5-17</td>
<td>Setting Up HNSG Parameter Area</td>
</tr>
<tr>
<td>5-18</td>
<td>Opening HNSG</td>
</tr>
<tr>
<td>5-19</td>
<td>Using IOCTL</td>
</tr>
<tr>
<td>5-21</td>
<td>Accessing HNSG From SYSC</td>
</tr>
</tbody>
</table>
HNSG: An Enhanced Serial Port Handler

Included in the HP-94 data communications pac is an external serial port handler, HNSG, with more capabilities than the built-in handler.

Features of HNSG

HNSG offers the following enhancements over the built-in handler:

- ENQ/ACK as well as XON/OFF pacing.
- Even/Odd/1's/0's parity with optional parity-checking and bad-character substitution.
- Selectable character delay.
- Supports IOCTL capability, including change-parameters (used by &CHCOMM) and BREAK (used by &SENBREAK).
- Enhanced error-handling.
- Buffer size is extended to 255 bytes.
- When XON/XOFF protocol is used, HNSG issues only one XOFF character when input buffer space is low; the built-in handler may issue several. HNSG will not issue another XOFF, even if the buffer overflows, until it has issued an XON.
- HNSG will not hang during a write request if CTS is low.

Installing HNSG

HNSG is contained on the Datacomm Pac disc in two files:

- HNSG.EXE can be incorporated into your application as a type "H" file in HXC.
- HNSG.MDS can be copied directly into an HP-94 with the HXCOPY utility.
HNSG Parameter Bytes

HNSG requires 7 parameter bytes. The following sections in this chapter will explain how to set up the parameters from BASIC, assembly-language, and SYSC. This table explains the meaning of each of the bytes.

<table>
<thead>
<tr>
<th>Byte #</th>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Always set to 254.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Baud rate: 9600 baud</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4800 baud</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2400 baud</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1200 baud</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>600 baud</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>300 baud</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>150 baud</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>No flow control</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>XON/XOFF flow control</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7 data bits</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>8 data bits</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>No parity</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Hardware parity control</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Odd parity</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Even parity</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1 stop bit</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2 stop bits</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>Do not strip incoming nulls</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Strip incoming nulls</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>Standard 3 bytes of parameters</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Extended parameters defined *</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Terminate code. If non-zero, this byte is used as a terminate character on send, and interpreted as a terminate character on receive.</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Extended flags. This byte is expected if bit 6 is set in the second byte.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Normal datacomm error reporting</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Delayed datacomm error reporting *</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Normal transmit error-handling</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Ignore datacomm errors on transmit *</td>
</tr>
</tbody>
</table>

* Items marked with an (*) are enhancements over the built-in handler.
### Meaning of Parameter Bytes for HNSG (cont.)

<table>
<thead>
<tr>
<th>Byte #</th>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>No ENQ/ACK pacing</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>ENQ/ACK pacing *</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Normal hardware-reported parity error handling</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Ignore hardware-reported parity errors *</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Normal hardware-reported overrun error handling</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Ignore hardware-reported overrun errors *</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>Normal hardware-reported framing error handling</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Ignore hardware-reported framing errors *</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Not defined — set to 0</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Not defined — set to 0</td>
</tr>
</tbody>
</table>

#### Byte 5
Software parity flags. This byte is expected iff bit 6 is set in the second byte.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No software parity control</td>
</tr>
<tr>
<td>1</td>
<td>Use software parity control *</td>
</tr>
<tr>
<td>1</td>
<td>Don’t check parity of data received by HP-94 *</td>
</tr>
<tr>
<td>1</td>
<td>Check parity of data received by HP-94 *</td>
</tr>
<tr>
<td>2</td>
<td>Forced parity *</td>
</tr>
<tr>
<td>1</td>
<td>Even or odd parity *</td>
</tr>
<tr>
<td>3</td>
<td>If bit 2 is zero:</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>If bit 2 is one:</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Report software-detected parity errors *</td>
</tr>
<tr>
<td>1</td>
<td>Replace software-detected parity errors with FFH *</td>
</tr>
<tr>
<td>5</td>
<td>Not defined — set to 0</td>
</tr>
<tr>
<td>6</td>
<td>Not defined — set to 0</td>
</tr>
<tr>
<td>7</td>
<td>Not defined — set to 0</td>
</tr>
</tbody>
</table>

#### Byte 6
Character delay. This byte is expected iff bit 6 is set in the third byte. It specifies a minimum delay, in msec, to be applied before sending each character.*

* Items marked with an (*) are enhancements over the built-in handler.

### Detailed Description of HNSG Parameters

This section assumes some knowledge of the built-in serial port handler, and concentrates on the differences found in HNSG. (Note: Except as noted below, Bytes 1 thru 3 of HNSG’s parameters correspond to bytes 0 thru 2 of the built-in handler’s parameters.)

To elaborate:
Byte #0, whose value must be 254, identifies the parameters as belonging to HNSG. Each external handler is assigned a unique 'valid-data' byte to aid in proper interpretation of its parameters.

Byte #1, the baud rate, has the same meaning used by the built-in handler.

Byte #2, bits 0-5, have the same meaning used by the built-in handler. Note that if you select hardware parity control (bit 2), the parity processing is performed by the 82C51 communications chip. HNSG also allows software-controlled parity processing (see below).

Byte #2, bit 6: If this bit is set, HNSG expects a total of 7 parameter bytes. If this bit is clear, HNSG expects four bytes: the 'valid-data' byte followed by the first three parameter bytes — the same three bytes used by the built-in driver.

Byte #2, bit 7: This bit is ignored when the device is opened, but has meaning when the parameters are changed. As explained in appendix F, HNSG will reset the 82C51 chip only if necessary when changing parameters. If this bit is set, HNSG will always reset the chip when changing parameters.

Byte #3, the terminate code, has the same meaning used by the built-in handler.

Bytes #4-6 are expected by HNSG only if bit 6 was set in byte #2. If bit 6 is not set, HNSG assumes a value of 0.

Byte #4, bit 0, provides an alternate form of error reporting:

- If this bit is clear, HNSG reports errors in the same way used by the built-in handler. To illustrate: if 40 characters are received and buffered, and a 41st character causes an error (parity, framing, or whatever), the next read will report an error immediately — before reading the first 40 characters from the buffer. A subsequent read is required to receive the buffered characters.

- If this bit is set, HNSG uses a 'delayed' error reporting scheme in which, in the above scenario, the error will not be reported until the 40 good characters have been read. Until the buffered characters are read, HNSG ignores all further incoming characters. The pending error (and the resulting lock against incoming data) is cleared when the input buffer is flushed through the IOCTL input-buffer-flush command.

Byte #4, bit 1, provides two alternatives in error reporting during a write request:

- If this bit is clear, HNSG defaults to behavior used in the built-in handler. If an error has occurred due to incoming data (parity error, framing error, buffer overflow, etc.), the error will be reported at the next I/O request: read or write.

- If this bit is set, HNSG will not report a receive error when it is handling a write request. The error will be reported when the next read request is issued.

Byte #4, bit 2, selects terminal-mode ENQ/ACK handshaking. After the HP-94 receives an <ENQ> character from the host, it will respond with an <ACK> character:

- Immediately, if the receive buffer is empty, or

- As soon as the receive buffer is emptied by a read.

If the host sends anything while an <ACK> is pending (host ENQ timeout), HNSG will cancel the...
pending transmission of <ACK>.

**Byte #4, bits 3-5**, instruct HNSG to ignore the three types of receive errors (parity, overrun, and framing) that can be detected by the 82C51 UART chip. If the hardware detects an error that is being ignored, the bad character is discarded and the error is not reported.

**Byte #5** instructs HNSG to control parity in software rather than in the 82C51 communications chip. This offers more choices of parity processing options.

**Byte #5, bit 0**, selects software parity control. If this bit is set, the 82C51 should be set to 8-bit no parity (in byte #2). If this bit is clear, the remaining bits in this byte are ignored – there is no software parity control.

**Byte #5, bit 1**, instructs HNSG to check the parity of incoming characters. If this bit is set, incoming characters will be checked against the current parity settings (bits 2-3) and handled according to the setting of bit 4. If this bit is clear, incoming characters are stripped of their high bit but not checked for parity.

**Byte #5, bits 2-3**, choose the type of parity:

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0's</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1's</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Odd</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Even</td>
</tr>
</tbody>
</table>

**Byte #5, bit 4**, selects the action to be performed when a parity error is detected in an incoming character (bit 1 must be set to perform parity-checking). If this bit is clear, the error will be reported just as with hardware parity-checking. If this bit is set, the bad character will be replaced with the byte FFH.

**Byte #8** selects a character delay. If this byte is non-zero, HNSG will delay this number of msec (1-255) before transmitting each character. The delay is approximate and might be lengthened by HP-94 interrupt activity.

---

**Accessing HNSG From BASIC**

The BASIC interface to HNSG consists of two parts:

- The SYSG utility, an expanded version of the built-in SYRS, that allows setting HNSG's parameters.
- The BASIC OPEN command, which actually opens the device handler.
The SYSG Utility

SYSG provides, for HNSG, the same function that SYRS provides for the built-in serial port handler. That is, it allows you to set the parameters that will be required by HNSG when channel #1 is opened. SYSG also offers capabilities that fully exploit the functions of the handler.

The following pages describe the methods of calling SYSG to access its various functions.
Setting OPEN Parameters with SYSG

$CALL SYSG(0,1,0,1,0,0,0,0)$
$CALL SYSG(0,4,1,6,0,0,3,0,"&0D")$

Description

$CALL SYSG(0,\ldots)$ sets the communications parameters to be used for the serial port interface when the interface is subsequently opened with the HNSG handler. The call must take place before opening the port – failure to do so will cause the port to be opened to the default parameters: 9600 baud, even parity, and 1 stop bit. If this call is made again after the device is opened, it will fail with an error #106 (channel already open).
### BAUD Specifier

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9600</td>
</tr>
<tr>
<td>2</td>
<td>4800</td>
</tr>
<tr>
<td>3</td>
<td>2400</td>
</tr>
<tr>
<td>4</td>
<td>1200</td>
</tr>
<tr>
<td>5</td>
<td>600</td>
</tr>
<tr>
<td>6</td>
<td>300</td>
</tr>
<tr>
<td>7</td>
<td>150</td>
</tr>
</tbody>
</table>

### HANDSHAKE Specifier

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Handshake</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No handshaking</td>
</tr>
<tr>
<td>1</td>
<td>XON/XOFF handshaking</td>
</tr>
<tr>
<td>2</td>
<td>ENQ/ACK handshaking</td>
</tr>
<tr>
<td>3</td>
<td>XON/XOFF and ENQ/ACK</td>
</tr>
</tbody>
</table>

### FORMAT Specifier

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Word Length (bits)</th>
<th>Stop Bits</th>
<th>Parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>1</td>
<td>none</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>1</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1</td>
<td>odd</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>odd</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>1</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>1</td>
<td>none</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>1</td>
<td>even</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1</td>
<td>even</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>2</td>
<td>none</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>2</td>
<td>none</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>2</td>
<td>odd</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>2</td>
<td>odd</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>2</td>
<td>none</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>2</td>
<td>none</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>2</td>
<td>even</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>2</td>
<td>even</td>
</tr>
</tbody>
</table>
Note

The FORMAT specifier selects what type of hardware parity control will be used. If software parity control is to be used (see SW_PARITY, below), the FORMAT specifier should be set to 1, 5, 9 or 13: that is; 8 bits, no parity.

### NULLSTRIP Specifier

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Null Strip</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>null strip disabled</td>
</tr>
<tr>
<td>1</td>
<td>null strip enabled</td>
</tr>
</tbody>
</table>

### Software Parity Control

If software parity control is used, the hardware parity control (FORMAT specifier) should be set for 8 bits, no parity.

### SW_PARITY Specifier

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Parity</th>
<th>Check Input Parity</th>
<th>Substitute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any even value</td>
<td></td>
<td>No software parity control</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0's</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>0's</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>Odd</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>Odd</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>1's</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>11</td>
<td>1's</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>13</td>
<td>Even</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>15</td>
<td>Even</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>17</td>
<td>0's</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>19</td>
<td>0's</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>21</td>
<td>Odd</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>23</td>
<td>Odd</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>25</td>
<td>1's</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>27</td>
<td>1's</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>29</td>
<td>Even</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>31</td>
<td>Even</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Parity. Four parity settings are available: 0's, 1's, even, and odd.

Check Input Parity. If this option is selected, input data is examined against the current parity setting. Data that does not match the current parity setting is handled as a parity error.

If this option is not selected, input data is stripped of its high bit but not checked against the current parity setting.

Substitute. If this option is selected along with the previous option (check input parity), bytes with parity errors are replaced with the byte $\text{FF}_{16}$.

If this option is not selected, but the check input parity option is, bytes with parity errors result in a read error (just as occurs with hardware parity control).

CDELAY

This parameter specifies a character delay (in msec) to be applied before transmitting each character.

SPECIAL Specifier

The various bits of this specifier select the special features of the HNSG handler.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Special Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Delayed read error reporting</td>
</tr>
<tr>
<td>1</td>
<td>Don't report read errors on write</td>
</tr>
<tr>
<td>2</td>
<td>Force reset</td>
</tr>
<tr>
<td>3</td>
<td>Ignore hardware-reported parity errors</td>
</tr>
<tr>
<td>4</td>
<td>Ignore hardware-reported overrun errors</td>
</tr>
<tr>
<td>5</td>
<td>Ignore hardware-reported framing errors</td>
</tr>
</tbody>
</table>

Delayed Read Error Reporting. This selects the delayed error reporting feature described in the "Detailed Description of HNSG Parameters" section, above.

Don't Report Read Errors on Write. If this feature is selected, errors resulting from incoming data are not reported when a write request is issued. See the "Detailed Description of HNSG Parameters" section, above.
**Force Reset.** This feature has no meaning for OPEN, but is used when changing parameters with SYSG. See the "Detailed Description of HNSG Parameters" section, above.

**Ignore Hardware-Reported Errors.** These bits instruct HNSG to ignore certain errors that can be detected by the 82C51 UART. If a character is received with an error that is to be ignored, the character is discarded and no error is reported.

**Example**

To select delayed error reporting and to ignore framing errors, select bits 0 and 5 for a value of 33.

**TERMCHAR Specifier**

The optional terminate character specifies what character will signal the end of incoming data, thereby allowing variable length data input. The terminate character can be any character except null (ASCII 0); specifying the null character or a null string is equivalent to having no terminate character.

The terminate character is also appended to each item output by PRINT # and PRINT #...USING. The optional end-of-line sequence sent by PRINT # and PRINT #...USING (<CR>/<LF>) is considered a separate item, so a terminate character is sent after that sequence as well.
Changing Parameters with SYSG

%CALL SYSG(1,1,0,1,0,0,0,0)
%CALL SYSG(1,4,1,6,0,0,3,0,"&0D")

Examples

Description

%CALL SYSG(1,...) sets the communications parameters for channel #1 when the channel is already opened with the HNSG handler.

Parameters are identical to those used for the %CALL SYSG(0,...) call (above).
Reading Current Parameters with SYSG

%CALL SYSG (2, Baud, Hshake,Fmt, Nulls, Swpar, Cdly, Spec, Tc$)

Example

%CALL SYSG(2,Baud,Hshake,Fmt,Nulls,Swpar,Cdly,Spec,Tc$)

Description

%CALL SYSG(2,...) returns the current value of the communications parameters into the passed variables.

The parameters have the same meanings as for %CALL SYSG(0,...) (above).
Sending a BREAK with SYSG

Example

%CALL SYSG(3)

Description

This function causes the HP-94 to send a BREAK to the host; the serial port transmit line is held in a "space" state for approximately 200 msec.
Reading the Buffer Status with SYSG

\%CALL SYSG(4, Numchars)

Example

\%CALL SYSG(4, Numchars)

Description

This function returns into the STATUS variable the number of characters currently in the input buffer.
Flushing the Input Buffer with SYSG

Example

%CALL SYSG(5)

Description

This function causes the HNSG handler to flush its input buffer. Also, if an error is pending due to the "delayed error-reporting" option, the pending error will be cancelled.
Opening HNSG

HNSG is opened using the standard OPEN call:

OPEN #1,"HNSG"

HNSG can be opened only to channel #1, and the open should be preceded by a call to SYSG (0,...) to set up the parameters. If parameters have not been set up with a call to SYSG, HNSG will be opened with the following default parameters: 7-bit, even parity, 1 stop bit, no terminating characters, no special features, no software parity control, no character delay.

Accessing HNSG From Assembly Language

Opening HNSG from assembly-language code consists of two parts:

- Setting up a parameter area, and
- Opening the channel.

The following sections also discuss how to use the IOCTL capabilities of HNSG. Full instructions on using IOCTL for device handlers can be found in the HP-94 Technical Reference Manual.

Setting Up HNSG Parameter Area

The operation of properly setting up the handler's parameter area is performed in BASIC by SYSG and in SYSC by the &OPEN command. This section details the work that must be performed by an assembly-language program to set up this area.

By way of background (detailed in the HP-94 Technical Reference Manual), the following code will load the HP-94 'monitor data segment' (MONIDS) into DS:

```
mov ax,16h
mov ds,ax ; DS points at system pointer table
mov ds,ds:[0] ; DS=MONIDS
```

Within the monitor data segment, the word at MONIDS:2 (ds:[2] after the above code is run) serves a double purpose:

- When channel #1 is open, the handler keeps its work area segment address in MONIDS:2.
- When channel #1 is closed, MONIDS:2 = 0 (if no parameters have been set up) or it points to a 1-paragraph area of allocated memory (allocated with system GETMEM function) containing the parameters needed for the OPEN operation. All external channel 1 handlers that use MONIDS:2 when open will restore this pointer when the channel is closed.

The following code segment illustrates setting up a parameter area for use by HNSG. Note that the
code needs to ensure that channel 1 is closed to guarantee that that MONIDS:2 contains the proper pointer.

In this example, the serial port is set up to be opened at 9600 baud with software-controlled even parity and ENQ/ACK handshaking (see the table at the beginning of this chapter for the meanings of the various bytes):

```
mov ax,16*256+1 ; Function 16, channel 1
int 1ah ; Ensure that serial device is closed
mov ax,16h ; Point DS at MONIDS
mov ds,ax
mov ds,ds:[0] ; Parm area allocated?
mov cx,ds:[2] ; Yes.
test ex,cx
jnz set_open_parms ; Get 1 paragraph
mov ax,11*256
mov bx,1
int 1ah
mov ds: [2]
mov byte ptr ds:[0],254
mov byte ptr ds:[1],1
mov byte ptr ds:[2],42h
mov byte ptr ds:[3],0
mov byte ptr ds:[4],4
mov byte ptr ds:[5],00h
mov byte ptr ds:[6],0

allocate_worked: ; Set MONIDS[2] to point here
mov ds:[2],cx

set_open_parms: ; Set DS at parameter area
mov ds,cx
mov byte ptr ds:[0],254
mov byte ptr ds:[1],1
mov byte ptr ds:[2],42h
mov byte ptr ds:[3],0
mov byte ptr ds:[4],4
mov byte ptr ds:[5],00h
mov byte ptr ds:[6],0

Opening HNSG

Once the parameter area is set up, channel 1 can be opened to HNSG using the system OPEN call. Since HNSG expects its parameters to be in the memory block pointed to by MONIDS:2, it does not use DS:DX as a parameter pointer, as does the built-in handler:

5-18 HNSG: An Enhanced Serial Port Handler
Using IOCTL

The IOCTL function allows a handler such as HNSG to offer extended functions not available through the standard OPEN/CLOSE/ etc. system calls. The use of IOCTL is explained more fully in the HP-94 Technical Reference Manual. The IOCTL functions supported by HNSG are:

<table>
<thead>
<tr>
<th>Function</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify</td>
<td>AH=00H</td>
<td>AL=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BX=&quot;HP&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CX=&quot;SG&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DH=Rev. #</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DL=Ver. #</td>
</tr>
<tr>
<td>Get Configuration</td>
<td>AH=01H</td>
<td>AL=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ES:DX-&gt;bytes</td>
</tr>
<tr>
<td>Change Configuration</td>
<td>AH=02H</td>
<td>AL=error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ES:DX-&gt;bytes</td>
</tr>
<tr>
<td>Buffer Status</td>
<td>AH=03H</td>
<td>AL=error code*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CX=# bytes</td>
</tr>
<tr>
<td>Flush Input Buffer</td>
<td>AH=04H</td>
<td>AL=0</td>
</tr>
<tr>
<td>Send BREAK</td>
<td>AH=81H</td>
<td>AL=0</td>
</tr>
</tbody>
</table>

* If there is an error that would be reported in the next READ or WRITE, it is reported in the Buffer Status IOCTL command and the error condition is cleared.
The following code, when near-called at CALL_IOCTL, will perform an IOCTL call to the channel 1 driver:

```
.driver_ioctl  proc far
  ER_CH_NOT_OPEN EQU 105
  OBIT EQU 10h
  call_ioctl label near
pushf
push bp
push es
push ds
push di
push si
push dx
push cx
push bx
push ax
mov bp,sp
mov bx,16h
mov ds,bx
mov bx,ds:[24h]
mov ds,ds:[0h]
add bx,12
test byte ptr ds:[bx],OBIT
mov al,ER_CH_NOT_OPEN
jz ioctl_done
xor ax,al
push cs
mov cx,offset ioctl_done
push cx
push de:[bx+3]
mov cx,de:[bx+1]
add cx,15h
push cx
mov ds,de:[bx+5]
mov cx,4[bp]
mov bx,2[bp]
ret ; Go to IOCTL handler

.driver_ioctl endp

ioctl_done proc near
mov [bp],al
pop ax
pop bx
pop cx
pop dx
pop si
pop di
pop ds
pop es
pop bp
popf
ret
ioctl_done endp
```

5-20 HNSG: An Enhanced Serial Port Handler
For example, to send a BREAK:

\[
\begin{align*}
\text{mov} & \quad \text{sh}, 81h \\
\text{call} & \quad \text{call_ioctt}
\end{align*}
\]

Accessing HNSG From SYSC

The SYSC commands &OPEN, &CHCOMM, and &SENDBREAK provide access to HNSG. For example, to open HNSG for 9600 baud, software-controlled even parity with ENQ/ACK handshaking (same parameters as in assembly-language example, above):

\[
&OPEN \quad "HNSG", 254, 1, \42, 0, 4, \0D, 0
\]

To change parameters to 1200 baud, XON/XOFF handshaking with no parity control:

\[
&CHCOMM \quad 254, 1, 2, 0
\]

Note that, since the extended-parameters bit is not set in byte #2, the last three bytes are not specified.

To send a BREAK:

\[
&SENDBREAK
\]
Error Conditions

SYXM Soft Errors

The following error codes will be displayed in the "code" field to indicate that a recoverable error has occurred:

<table>
<thead>
<tr>
<th>Number</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NAK received</td>
</tr>
<tr>
<td>2</td>
<td>Unexpected character received</td>
</tr>
<tr>
<td>3</td>
<td>Incorrect checksum or CRC</td>
</tr>
<tr>
<td>4</td>
<td>First character not SOH</td>
</tr>
<tr>
<td>5</td>
<td>No response received within timeout period</td>
</tr>
<tr>
<td>6</td>
<td>Short packet received</td>
</tr>
<tr>
<td>7</td>
<td>Invalid block count</td>
</tr>
<tr>
<td>8</td>
<td>Duplicate block ignored</td>
</tr>
</tbody>
</table>

Datacomm Fatal Errors

The following errors can occur while using the HP-94 datacomm pac. Most of the errors are standard HP-94 errors. Errors marked with a (*) are SYSC errors. Those marked with a (†) are SYXM errors. A (†) indicates an SYSG error.
### Error Conditions That Can Occur in the HP-94 Datacomm Pac

<table>
<thead>
<tr>
<th>Number</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Illegal parameter</td>
</tr>
<tr>
<td>102</td>
<td>Invalid directory number</td>
</tr>
<tr>
<td>103</td>
<td>File not found</td>
</tr>
<tr>
<td>104</td>
<td>Too many files</td>
</tr>
<tr>
<td>105</td>
<td>Channel not open</td>
</tr>
<tr>
<td>106</td>
<td>Channel already open</td>
</tr>
<tr>
<td>107</td>
<td>File already open</td>
</tr>
<tr>
<td>108</td>
<td>File already exists</td>
</tr>
<tr>
<td>109</td>
<td>Read-only access</td>
</tr>
<tr>
<td>110</td>
<td>Access restricted</td>
</tr>
<tr>
<td>111</td>
<td>No room for file</td>
</tr>
<tr>
<td>112</td>
<td>No room to expand file</td>
</tr>
<tr>
<td>113</td>
<td>No room for scratch area</td>
</tr>
<tr>
<td>116</td>
<td>Terminate character detected</td>
</tr>
<tr>
<td>118</td>
<td>Timeout</td>
</tr>
<tr>
<td>119</td>
<td>Power switch pressed</td>
</tr>
<tr>
<td>150*</td>
<td>Cannot parse script command</td>
</tr>
<tr>
<td>151*</td>
<td>Byte parameter out of 0-255 range</td>
</tr>
<tr>
<td>152*</td>
<td>Cannot parse parameter or Wrong parameter type</td>
</tr>
<tr>
<td>153*</td>
<td>Out of stack space</td>
</tr>
<tr>
<td>154*</td>
<td>Label not found</td>
</tr>
<tr>
<td>155*</td>
<td>Serial handler does not support IOCTL operations</td>
</tr>
<tr>
<td>156‡</td>
<td>IOCTL operation attempted on the wrong handler</td>
</tr>
<tr>
<td>160†</td>
<td>Cancel requested by host computer</td>
</tr>
<tr>
<td>161†</td>
<td>Incorrect block count</td>
</tr>
<tr>
<td>200</td>
<td>Low main battery voltage</td>
</tr>
<tr>
<td>201</td>
<td>Receive buffer overflow</td>
</tr>
<tr>
<td>202</td>
<td>Parity error</td>
</tr>
<tr>
<td>203</td>
<td>Overrun error</td>
</tr>
<tr>
<td>204</td>
<td>Parity and overrun error</td>
</tr>
<tr>
<td>205</td>
<td>Framing error</td>
</tr>
<tr>
<td>206</td>
<td>Framing and parity error</td>
</tr>
<tr>
<td>207</td>
<td>Framing and overrun error</td>
</tr>
<tr>
<td>208</td>
<td>Framing, overrun, and parity error</td>
</tr>
<tr>
<td>218</td>
<td>Lost connection while transmitting</td>
</tr>
</tbody>
</table>