HP-94 Handheld Industrial Computer

Technical Reference Manual Addendum

HEWLETT PACKARD

Edition 1 May 1987

Reorder Number
82521-90003

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.

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

Printing History **Edition 1** May 1987 Mfg. No. 82521-90003

Contents

Technical Reference Manual Addendum

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).
- Errors in the *HP-94 Technical Reference Manual* (Edition 1).
- Additional information for the *Technical Reference Manual.*

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:

  ```
  Copr. HP 1985 Vx.xx
  ```

  The `x.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 CR LF 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 CR LF cannot be transmitted. When the program tries to send the CR LF 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:

```
                cmp     al,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
                                            ;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
```

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
                    .
                    .
                    .
XON                 db      11h               ;XON
                    .
                    .
                    .
```

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.

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                  ;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 clear the memory area
                .
                .
                .
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:

```
                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      "SYPA"              ;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.

```
.
.
.
mov         si,16h              ;Get operating system pointer table
mov         ds,si
mov         si,ds:[14h]         ;Get offset of status area
mov         ds,ds:[00h]         ;DS = OS scratch space
mov         al,ds:[si+02h]      ;Get saved copy of main control register
mov         ds:[16h],al         ;Place it in OS scratch space
.
.
.
```

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.

```
.
.
.
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.

IOCTL Function	Workaround
CHANGE_CONFIG (02h)	Close high- and low-level handler pair Use SYBC or SYSP to change low-level handler configuration Open high- and low-level handler pair
GET_CONFIG (01h)	Close high- and low-level handler pair Open low-level handler Use XIOCTL utility to call GET_CONFIG function Close low-level handler Open high- and low-level handler pair
IDENTIFY (00h)	Close high- and low-level handler pair Open low-level handler Use XIOCTL utility to call IDENTIFY function Close low-level handler Open high- and low-level handler pair
RECEIVE_FLUSH (04h)	Close high- and low-level handler pair Open high- and low-level handler pair
RECEIVE_STATUS (03h)	Do not use HNWN with the low-level handler
WR_RD_EN (80h) (HNSP only)	Close high- and low-level handler pair Open low-level handler Use XIOCTL utility to call WR_RD_EN function Close low-level handler Open high- and low-level handler pair

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 ␍␊.

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, EVENTTBL, 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:

Old Name	New Name
KEY_SCAN	KEY_SCAN_JMP
CURSOR_BLINK	CURSOR_BLINK_JMP
TIMEOUT	TIMEOUT_JMP
VERSION	ROM_VERSION

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

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

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

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

HP-94 Console Status

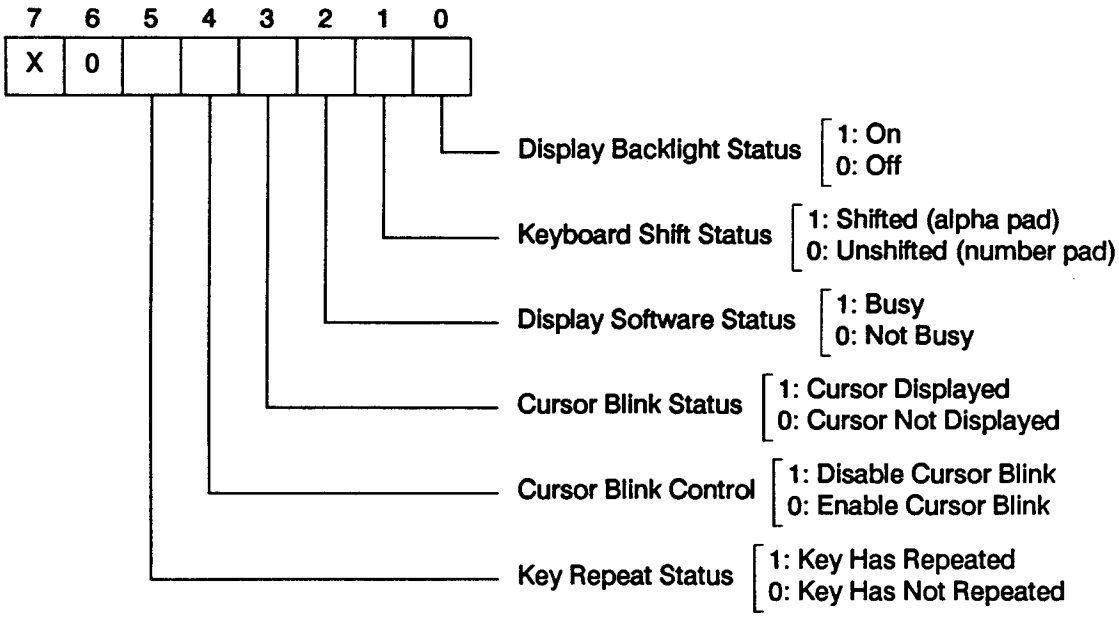
A small modification of the SCANKYBD utility provided on the *Technical Reference Manual* disc will allow determining six aspects of HP-94 console status:

- 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:



X = ignore

Console Status Byte Returned by Modified SCANKYBD Utility

5

HNSG: An Enhanced Serial Port Handler

Contents

Chapter 5 HNSG: An Enhanced Serial Port Handler

5

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 &SENDBREAK).
- 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.

Meaning of Parameter Bytes for HNSG

Byte #	*Bit*	*Meaning*
0		Always set to 254.
1		Baud rate: 1 9600 baud 2 4800 baud 3 2400 baud 4 1200 baud 5 600 baud 6 300 baud 7 150 baud
2	0	0 No flow control 1 XON/XOFF flow control
	1	0 7 data bits 1 8 data bits
	2	0 No parity 1 Hardware parity control
	3	0 Odd parity 1 Even parity
	4	0 1 stop bit 1 2 stop bits
	5	0 Do not strip incoming nulls 1 Strip incoming nulls
	6	0 Standard 3 bytes of parameters 1 Extended parameters defined *
	7	Only meaningful when changing parameters: 0 Do not reset 82C51 unless format changes * 1 Force reset of 82C51 *
3		Terminate code. If non-zero, this byte is used as a terminate character on send, and interpreted as a terminate character on receive.
4		Extended flags. This byte is expected iff bit 6 is set in the second byte.
	0	0 Normal datacomm error reporting 1 Delayed datacomm error reporting *
	1	0 Normal transmit error-handling 1 Ignore datacomm errors on transmit *
* Items marked with an (*) are enhancements over the built-in handler.		

Meaning of Parameter Bytes for HNSG (cont.)

Byte #	*Bit*	*Meaning*
	2	0 No ENQ/ACK pacing 1 ENQ/ACK pacing *
	3	0 Normal hardware-reported parity error handling 1 Ignore hardware-reported parity errors *
	4	0 Normal hardware-reported overrun error handling 1 Ignore hardware-reported overrun errors *
	5	0 Normal hardware-reported framing error handling 1 Ignore hardware-reported framing errors *
	6	Not defined — set to 0
	7	Not defined — set to 0
5		Software parity flags. This byte is expected iff bit 6 is set in the second byte.
	0	0 No software parity control 1 Use software parity control *
	1	0 Don't check parity of data received by HP-94 * 1 Check parity of data received by HP-94 *
	2	0 Forced parity * 1 Even or odd parity *
	3	If bit 2 is zero: 0 0's parity * 1 1's parity * If bit 2 is one: 0 Odd parity * 1 Even parity *
	4	0 Report software-detected parity errors * 1 Replace software-detected parity errors with FF_H *
	5	Not defined — set to 0
	6	Not defined — set to 0
	7	Not defined — set to 0
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:

Bit 2	Bit 3	Parity
0	0	0's
0	1	1's
1	0	Odd
1	1	Even

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 FF_H.

Byte #6 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:

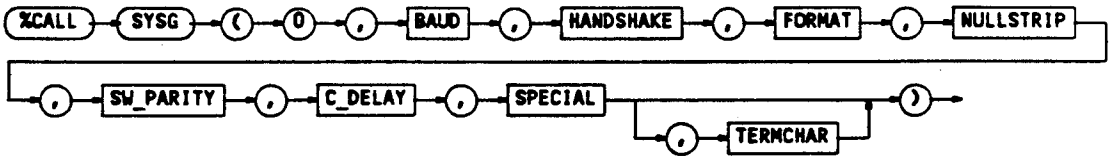
- The SYSG utility, an expanded version of the built-in SYRS, that allows setting HNSG's parameters.
- The BASIC OPEN command, which actually opens the device handler.

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



Item	Description	Range
BAUD	numeric expression, truncated to an integer	1 through 7
HANDSHAKE	numeric expression, truncated to an integer	0 through 3
FORMAT	numeric expression, truncated to an integer	0 through 15
NULLSTRIP	numeric expression, truncated to an integer	0 through 1
SW_PARITY	numeric expression, truncated to an integer	0 through 31
C_DELAY	numeric expression, truncated to an integer	0 through 255
SPECIAL	numeric expression, truncated to an integer	0 through 63
TERMCHAR	string expression consisting of one character	-

Examples

```
%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,...)` 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

Specifier	Baud Rate
1	9600
2	4800
3	2400
4	1200
5	600
6	300
7	150

HANDSHAKE Specifier

Specifier	Handshake
0	No handshaking
1	XON/XOFF handshaking
2	ENQ/ACK handshaking
3	XON/XOFF and ENQ/ACK

FORMAT Specifier

Specifier	Word Length (bits)	Stop Bits	Parity
0	7	1	none
1	8	1	none
2	7	1	odd
3	8	1	odd
4	7	1	none
5	8	1	none
6	7	1	even
7	8	1	even
8	7	2	none
9	8	2	none
10	7	2	odd
11	8	2	odd
12	7	2	none
13	8	2	none
14	7	2	even
15	8	2	even

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

Specifier	Null Strip
0	null strip disabled
1	null strip enabled

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

Specifier	Parity	Check Input Parity	Substitute
Any even value	No software parity control		
1	0's	no	no
3	0's	yes	no
5	Odd	no	no
7	Odd	yes	no
9	1's	no	no
11	1's	yes	no
13	Even	no	no
15	Even	yes	no
17	0's	no	no
19	0's	yes	yes
21	Odd	no	no
23	Odd	yes	yes
25	1's	no	no
27	1's	yes	yes
29	Even	no	no
31	Even	yes	yes

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 FF_H.

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.

SPECIAL Specifier

Bit	Special Feature
0	Delayed read error reporting
1	Don't report read errors on write
2	Force reset
3	Ignore hardware-reported parity errors
4	Ignore hardware-reported overrun errors
5	Ignore hardware-reported framing errors

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

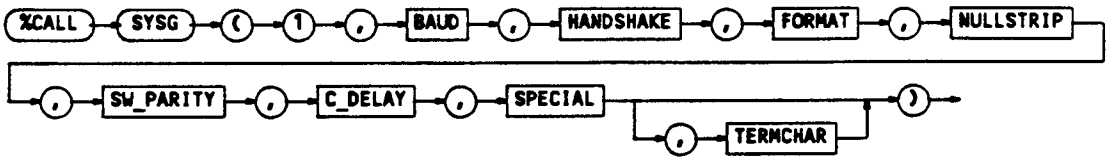
To select delayed error reporting and to ignore framing errors, select bits 0 and 5 for a value of 33.

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



Item	Description	Range
BAUD	numeric expression, truncated to an integer	1 through 7
HANDSHAKE	numeric expression, truncated to an integer	0 through 3
FORMAT	numeric expression, truncated to an integer	0 through 15
NULLSTRIP	numeric expression, truncated to an integer	0 through 1
SW_PARITY	numeric expression, truncated to an integer	0 through 31
C_DELAY	numeric expression, truncated to an integer	0 through 255
SPECIAL	numeric expression, truncated to an integer	0 through 7
TERMCHAR	string expression consisting of one character	–

Examples

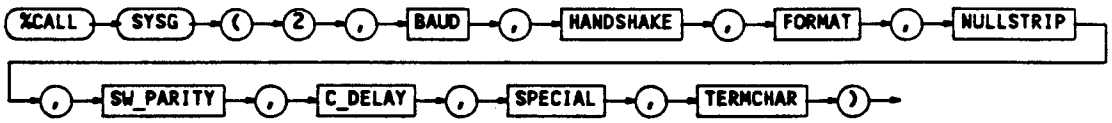
```
%CALL SYSG(1,1,0,1,0,0,0,0)
%CALL SYSG(1,4,1,6,0,0,3,0,"&0D")
```

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



Item	Description
BAUD	numeric variable
HANDSHAKE	numeric variable
FORMAT	numeric variable
NULLSTRIP	numeric variable
SW_PARITY	numeric variable
C_DELAY	numeric variable
SPECIAL	numeric variable
TERMCHAR	string variable

Example

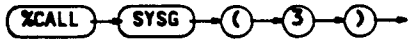
```
%CALL SYSG(2,Baud,Hshake,Fmt,Nulls,Swpar,Cdly,Spec,Tc$)
```

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

Item	Description
STATUS	numeric variable

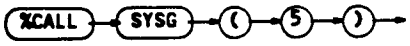
Example

```
%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
                mov     ds,ax
                mov     ds,ds:[0]               ; Point DS at MONIDS
                mov     cx,ds:[2]
                test    cx,cx                   ; Parm area allocated?
                jnz     set_open_parms          ; Yes.
                mov     ax,11*256
                mov     bx,1
                int     1ah                     ; Get 1 paragraph
                test    al,al
                jz      allocate_worked         ; Go if worked
                jmp     error_report            ; Report failure to create work area
allocate_worked:
                mov     ds:[2],cx               ; Set MONIDS[2] to point here
set_open_parms:
                mov     ds,cx                   ; Point DS at parameter area
                mov     byte ptr ds:[0],254     ; Set valid-data byte
                mov     byte ptr ds:[1],1       ; Baud rate = 9600
                mov     byte ptr ds:[2],42h     ; 8-bits, no parity, expect extended parms
                mov     byte ptr ds:[3],0       ; No terminating character
                mov     byte ptr ds:[4],4       ; ENQ/ACK handshaking
                mov     byte ptr ds:[5],00h     ; Even parity, don't check incoming chars
                mov     byte ptr ds:[6],0       ; No character delay
                .
                .
                .
```

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:

```
                push    cs
                pop     es
                mov     bx,offset hand_name          ; ES:BX -> handler name
                mov     ax,15*256+1                  ; Function 15, channel 1
                int     1ah                          ; Open channel 1 to HNSG
                .
                .
                .
hand_name       db      "HNSG",0
```

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:

Function	Input	Output
Identify	AH=00_H	AL=0 BX="HP" CX="SG" DH=Rev. # DL=Ver. #
Get Configuration	AH=01_H	AL=0 ES:DX→bytes
Change Configuration	AH=02_H ES:DX→bytes	AL=error code
Buffer Status	AH=03_H	AL=error code* CX=# bytes
Flush Input Buffer	AH=04_H	AL=0
Send BREAK	AH=81_H	AL=0

* 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]             ; OPNTBL
                mov     ds,ds:[0h]              ; MONIDS
                add     bx,12                   ; DS:BX points at chan 1 in OPNTBL
                test    byte ptr ds:[bx],OBIT   ; Is file open?
                mov     al,ER_CH_NOT_OPEN
                jz      ioctl_done
                xor     al,al
                push    cs
                mov     cx,offset ioctl_done
                push    cx                      ; Stack has rtn address for ioctl_done
                push    ds:[bx+3]
                mov     cx,ds:[bx+1]
                add     cx,15h                  ; Point to IOCTL vector in handler
                push    cx                      ; Stack has execute address for ioctl
                mov     ds,ds:[bx+5]            ; Load handler DS
                mov     cx,4[bp]
                mov     bx,2[bp]                ; Restore these registers
                ret     ; Go to IOCTL handler
driver_ioctl    endp

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

For example, to send a BREAK:

```
mov     ah,81h
call    call_ioctl
```

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 "HNSG",254,1,\42,0,4,\0D,0
```

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

```
&CHCOMM 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
```

L

Error Conditions

SYXM Soft Errors

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

Number	Condition
1	NAK received
2	Unexpected character received
3	Incorrect checksum or CRC
4	First character not SOH
5	No response received within timeout period
6	Short packet received
7	Invalid block count
8	Duplicate block ignored

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

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