HP Instant Support Corporate Edition

Hardware diagnostics

Introduction

HP Diagnostics Object Model (DOM) is a set of hardware diagnostic test modules that are used as part of HP Instant Support.

The available tests are:

- Processor Test
- Memory Test
- Audio Test
- Microphone Test
- Monitor Test
- PCI Bus Test
- Parallel Port Test
- Serial Port Test
- Video Test
- Floppy Drive Test
- CDROM Drive Test
- Hard Disk Drive (HDD) Test
- Keyboard Test
- Local Area Network (LAN) Test
- Modem Test
- Mouse Test
- USB Test

All tests are designed to be downloadable from the web, to run on the Microsoft operating systems (9X, NT4, 2K, XP) and to be compliant with the Distributed Management Task Force (DMTF) Common Diagnostics Model (CDM) standard.

Processor test

The processor diagnostic has four phases. Each one tests each operation (logical, arithmetic, floating point and MMX).

CPU logical operations

CPU logical operations: testing shall be conducted on the proper operation of the logical instructions (AND, OR, XOR, NOT, TEST, BT, BTC, BTR, BTS, BSF, BSR, SHL, SHR, SAL, SAR, ROL, ROR, RCL, RCR, SHLD, SHRD).

CPU arithmetic operations

CPU arithmetic operations: testing shall be conducted on the proper operation of the arithmetic instructions (ADD, ADC, SUB, SBB, CMP, INC, DEC, DIV, IDIV, MUL, IMUL, AAA, AAD, AAM, AAS, DAA, DAS).

Floating point operations

Testing shall be conducted on the proper operation of the floating point instructions (FBLD, FILD, FLD, FLD1, FLDL2E, FLDL2T, FLDLG2, FLDLN2, FLDPI, FLDZ, FBSTP, FIST, FISTP, FST, FSTP, F2XM1, FABS, FADD, FADDP, FIADD, FCHS, FCOM, FCOMP, FCOMPP, FICOM, FICOMP, FUCOM, FUCOMP, FUCOMPP, FCOS, FDIV, FDIVP, FIDIV, FDIVR, FDIVRP, FIDIVR, FMUL, FMULP, FIMUL, FPATAN, FPREM, FPREM1, FPTAN, FRNDINT, FSCALE, FSIN, FSINCOS, FSQRT, FSUB, FSUBP, FISUB, FSUBR, FSUBRP, FISUBR, FTST, FXAM, FXTRACT, FYL2X, FYL2XP1)

Action flow

The floating-point operation diagnostics shall have the following action flow (using 'C' language code, checking if the compiler generates floating point instructions):

- Test Pentium floating point bug: (4195835.0 / 3145727.0) * 3145727.0

  ```
  float val1 = 4195835.0;
  float val2 = 3145727.0;
  float x, y;

  x = val1 / val2;
  y = x * val2;

  if ( y != val1 )
  return ERROR;
  return OK;
  ```

- Make the following calculations, checking its results (use floating type variables):

  Using FSQRT, FADD, FMUL, FDIV, FSUB instructions, and considering PI = FLDPI (PI generated by the coprocessor) calculate:

  sqrt(PI) == 1.7724538509055403f

*PI == sqrt(PI) * sqrt(PI)*

*PI *3 == PI + PI + PI*

*PI / 3 == ((3 * PI) – (2 *PI)) /3*

Using FYL2X and FYL2XP1 instructions, calculate:

*7 * Log23 == 7 * Log2(2+1)*

Using FLDL2E, FLDL2T and FYL2X instructions, calculate:

Log2e == Log2e

Log210 == Log210

Using FLDZ and FTST instructions, calculate:

Push (0.0)

ST == 0

MMX operations

Testing shall be conducted on the proper operation of the MMX instructions (PADD, PADDS, PSUB, PSUBUS, PMUL, PMADDWD, PCMP, PACK, PUNPCK, PAND, PANDN, POR, PXOR, PSLL, PSRL, PSRA).

Memory test

General technical overview

The following sections and subsections explain precisely how the memory diagnostic works. In order to minimize risks, the diagnostic is not a driver based (kernel level) but an application. All tests are done in physical memory through Virtual Memory Manager.

All algorithms have been designed in order to have only one multi-operating system (OS) memory diagnostic for:

1. Win98 SE
2. Windows NT4.0 Workstation and Server (with at least SP4)
3. Windows 2000 Professional and Advanced Server
4. Windows XP

Because of the time execution of some subtests, HP DOM Memory diagnostic has two modes:

1. Quick (→ QuickMode of ParamTestSettings)
2. Advanced (→ Normal mode)

Diagnostic principle

The goal of the diagnostic is to test at least 80% of the physical memory. In order to accomplish this, the diagnostic must be able to fill memory banks without being swapped by the OS.

Main algorithm

The main algorithm of the memory diagnostic is as follows:

1. Check total physical memory size of the system
2. Check available physical memory of the system
3. Swap other processes in order to gain more available physical memory space
4. Once maximum physical memory space is reached:
    a. Return the size of the memory block allocated
    b. Free the allocated memory block
    c. Re-allocate only 90% of the available physical memory space
5. If quick tests mode is enabled, perform:
    a. Own Address Test
    b. Pattern Test
6. If Normal mode is enabled, perform:
    a. Tests of the quick tests mode
    b. Checker-Board Test
    c. Walking Zero's Test
    d. Walking One's Test
7. Complete the diagnostics by testing the 10% remaining physical memory space

a. At the same time, re-read memory pages corresponding to the memory block already tested to avoid being swapped

8. Re-allocate the 10% remaining physical memory space.

Memory tests algorithm – quick tests

Own address test

This test allows testing errors coming from the addresses decoder and shall catch any addressing errors.

Test principle

Each physical address is written with its corresponding virtual address and then is checked for consistency.

- From the lowest virtual address of the virtual memory block allocated
  - *While the highest virtual address of the virtual memory block is not reached do:*
    - *Write the virtual address (DWORD) in physical memory*

    End while

- From the lowest virtual address of the virtual memory block allocated
  - *While the highest virtual address of the virtual memory block is not reached do:*
    - *Read value stored at the virtual address (DWORD)*
    - *Verify that the value read matches with the virtual address.*

    End while

Pattern test

Test principle

The pattern test writes a series of nine different pattern sets in a physical memory block, and then reads the block back. Each pattern is a two byte (char) sized pattern and is toggled by their inverse after having verified an entire memory range.

Pattern	Inverse Pattern
0x55	0xAA
0xE1	0x1E
0xD2	0x2D
0xC3	0x3C
0xB4	0x4B
0xA5	0x5A
0x96	0x69
0x87	0x78
0xF0	0x0F

Table 1. "Pattern test" patterns

Memory tests algorithm – normal tests

Extended pattern test

This test is the same test (same algorithm) as the pattern test described in the Zero Pattern Test, with additional patterns being used: WORD and DWORD.

Pattern	Inverse Pattern
0x55, (BYTES) 0x5555, (WORD) 0x55555555 (DWORD)	0xAA, (BYTES) 0xAAAA, (WORD) 0xAAAAAAAA (DWORD)
0xE1, 0xE1E1, 0xE1E1E1E1	0x1E, 0x1E1E, 0x1E1E1E1E
0xD2, 0xD2 D2 0xD2 D2D2	0x2D, 0x2D2D, 0x2D2D2D2D
0xC3, 0xC3C3, 0xC3C3C3C3	0x3C, 0x3C3C, 0x3C3C3C3C
0xB4, 0xB4B4, 0xB4B4 B4B4	0x4B, 0x4B4B, 0x4B4B4B4B
0xA5, 0xA5A5, 0xA5A5A5A5	0x5A, 0x5A5A, 0x5A5A5A5A
0x96, 0x9696, 0x96969696	0x69, 0x6969, 0x69696969
0x87, 0x8787, 0x87878787	0x78, 0x7878, 0x78787878
0xF0, 0xF0F0, 0xF0F0F0F0	0x0F, 0x0F0F, 0x0F0F0F0F

Table 2. "Extended pattern test" patterns

Walking zero test

Test principle

The Walking Zero's test shall test and verify all address bits in all memory banks.

- From the lowest virtual address of the virtual memory block allocated
    - Fills a tested block of memory with 1's
    - Writes a walking 0's memory sequence to one location in a tested memory block
    - Checks to see if another memory element within the same tested block of memory has erroneously changed.

Walking one's test

The Walking 1's test is the same as Walking 0's but 0 is replaced by 1 and 1 by 0.

Limitations

Physical memory blocks

It is not possible to test all physical memory because the OS takes some physical memory blocks. Several investigations have proven that the maximum space reserved exclusively by the OS is approximately 30 MB (for Windows 2000 Professional). We assume the space reserved by the OS is valid and that it does not require testing.

Limited coverage

As it is impossible to manage physical memory directly, some memory tests offer limited coverage. Coverage is lost because a continuous virtual memory block is not mapped continuously in physical memory. The virtual block is generally cut in several physical memory blocks with different sizes. Nevertheless, the direction of reading is respected. Currently all Windows operating systems read and write in physical memory in the same way that it reads and writes in virtual memory.

Note: Because this test has the ability to swap memory, it is much more comprehensive than many standard on-line memory diagnostics (which do not have this capability).

Audio test

Implementation aspects

The audio diagnostic has three phases. The first phase shall test "wav", the second test "midi" and the third a simple speaker test.

- First phase: The "wav" phase is intended to test the basic sound functionality by playing the standard windows sound format (wav). This test is done using standard windows sound API. The sound consists of playing a wav sound in the left channel, in the right channel, and finally, in both channels. The user is prompted in order to confirm he hears the sound accordingly. The wav files played in this phase are integrated in the diagnostic resource file.

- Second phase: This phase shall test the basic "midi" capability of the sound card. It is responsible for playing a simple midi file. Again, the user is requested to confirm if he hears the sound accordingly.

- Third phase: This phase shall test the PC speaker basic functionality by producing a "beep" on the PC speaker.

Audio flow

WAV test

1. Display message box MSG_WAV_TEST
2. Display info message MSG_WAV
    a. If the sound device driver is not present, generate error ERROR_NO_SOUND_DRIVER and abort test
3. Play a stereo sound in both channels (left and right). The sound should be a *.wav sound stored in the resource file.
    a. If there is not enough memory to play the sound, generate error ERROR_NO_MEMORY and abort the wav test.
    b. If there was an error playing the sound, generate error ERROR_WAV_SYSTEM_BOTH and abort the wav test.
4. Prompt the user with message PROMPT_WAV_BOTH
    a. If the user answers "no", then generate error ERROR_WAV_BOTH
5. Play a sound in the left channel. The sound should be a *.wav sound stored in the resource file. It should be stored to be played in the left channel only.
    a. If there was an error playing the sound, generate error ERROR_WAV_SYSTEM_LEFT and abort the wav test.
6. Prompt the user with message PROMPT_WAV_LEFT
    a. If the user answers "No", then generate error ERROR_WAV_LEFT
7. Play a sound in the right channel. The sound should be a *.wav sound stored in the resource file. It should be stored to be played in the right channel only.
    a. If there was an error playing the sound, generate error ERROR_WAV_SYSTEM_RIGHT and abort the wav test.
8. Prompt the user with message PROMPT_WAV_RIGHT

a. If the users answer “No”, then generate error ERROR_WAV_RIGHT

MIDI test

1. Display message box MSG_MIDI_TEST
2. Display info message MSG_MIDI
3. Play a sound from a midi file. The sound should be heard from both channels.
   a. If there was an error playing the sound, generate error ERROR_MIDI_SYSTEM and abort the midi test
4. Prompt the user with message PROMPT_MIDI
   a. If the user answers “No”, then generate error ERROR_MIDI

PC speaker test

1. Display message box MSG_SPEAKER_TEST
2. Display info message MSG_SPEAKER
3. Play a beep in the PC Speaker.
   a. If there was an error playing the sound, then generate error ERROR_PCSPEAKER_SYSTEM and abort the test
4. Prompt the user with message PROMPT_PCSPEAKER
   a. If the user answers “No”, then generate error ERROR_PCSPEAKER

Microphone test

Implementation aspects

The microphone diagnostic has two phases. In the first phase, some sounds shall be recorded with the microphone; in the second phase, the sound shall be reproduced.

- First phase: this phase is intended to test the basic recording functionality by asking the user to produce some sound and recording this sound in a temporary file in "wav" format. This test is done using windows MCI.

- Second phase: this phase shall play the "wav" file recorded and request the user to confirm if he hears the sound accordingly. There shall be no verification of stereo or volume capabilities.

Action flow

Detecting mute

If mute was detected in the system or wav channel prior to the diagnostics starting, then a MSG_WARNING_WAV_MUTE shall be generated. If the microphone was set to mute, then the warning message MSG_WARNING_MIC_MUTE shall be generated.

If it fails to get microphone mute or to access volume control, the warning MSG_WARNING_MIC_SETTINGS must be generated.

When the diagnostic starts, if mute was detected in microphone, wav or system channels, a message box with warning message WARNING_TEST_MUTE shall be generated and stops recording and playing shall not be executed.

Recording phase

1. Check if there is a sound driver installed.
    a. If not, display error MSG_ERROR_NO_SOUND_DRIVER and abort the test.
2. Check if there is enough memory to open the sound driver.
    a. If not, display error MSG_ERROR_NO_MEMORY and abort the test.
3. If the waveform-audio device driver is no present, generate error ERROR_NO_AUDIO_DRIVER and abort the test.
4. Display message box MSG_REC_TEST
5. Try to open the device.
    a. If it doesn't succeed, generate error ERROR_OPEN_DEVICE and abort the test.
6. Try to record sound produced for five seconds.
    a. If it doesn't succeed, generate error ERROR_RECORD. There is no need to save the sound in a temporary file.
7. Display message box MSG_REC_TEST_FINISHED

Playing phase

1. Display message box MSG_REPR_TEST
    a. If there is not enough memory to play the sound, generate error ERROR_NO_MEMORY and abort the test.

b. If there was an error playing the sound, generate error ERROR_PLAY and abort the test.

2. Prompt the user with message PROMPT_WAV_REPRODUCED

   a. If the user answers "no", then generate error ERROR_PLAY_USER

User interface layout

The user interface for this diagnostic does not follow the generic template. The reason is the nature of the diagnostic that is inherently interactive and the existence of controls to manipulate the sound volume.

During the recording phase, the interface shall show a picture of a microphone and a label blinking with "recording…". It indicates that the diagnostic is expecting the user to use the microphone to record some sound. Volume and mute controls are also shown to let the user adjust the recording.

The pause button shall always be disabled, because of the nature of the diagnostic.

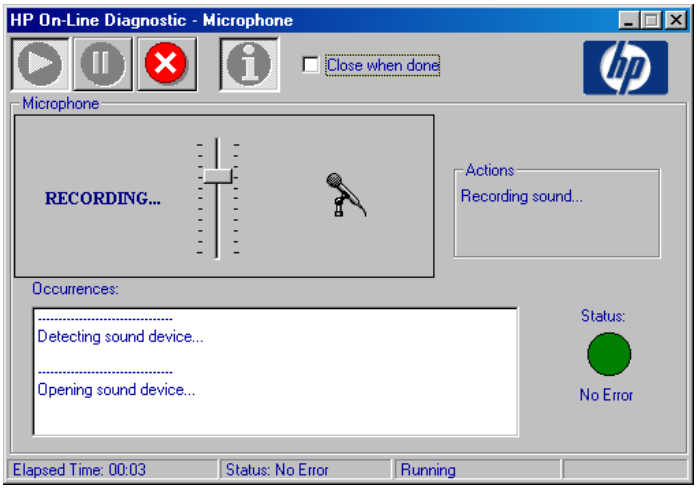

Figure 1. Microphone recording interface

During the reproducing phase, the interface shall show pictures of speakers and notes. It indicates that the recorded sound is being reproduced. Volume and mute controls are also shown to let the user adjust the reproduction.

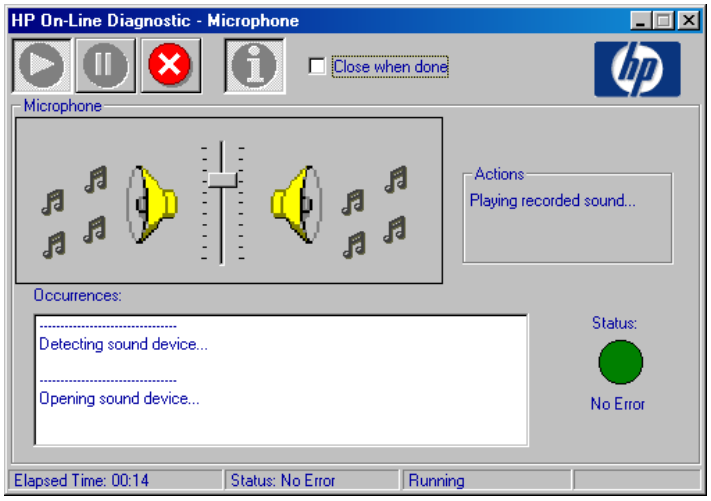

Figure 2. Microphone reproducing interface

Monitor test

Implementation aspects

The monitor diagnostic shall have five phases. The first phase shall be responsible for determining the supported resolution and colors supported by the video adapter and monitor. The second phase shall begin the first test, which shall be the resolution test. Third, the color test shall take place. Fourth, the focus test shall be run. Finally, there shall be the geometry test.

- First phase: This phase shall search the registry and Windows API looking for supported resolutions and colors.
- Second phase: This phase shall be responsible for the resolution test. The diagnostic shall attempt to change to each supported resolution in a valid refresh rate configuration. The user shall be prompted to respond if the screen was displayed correctly.
- Third phase: This phase shall be responsible for the color test. The diagnostic shall show a full screen with each basic color: black, white, red, blue, green. The user shall be prompted to respond to ensure that the colors were displayed correctly.
- Fourth phase: This phase shall be responsible for the focus test. It shall display a series of screens with thin lines. The user shall be prompted to respond if the screens were displayed correctly.
- Fifth phase: This phase shall be responsible for the geometry test. It shall show a template on the screen with geometric pictures (circles, squares and oblique lines). The user shall be prompted to respond if the screen was displayed correctly.

Algorithm

Variables, constants and structures

Name	Description
MIN_VERTICAL_RESOLUTION	Constant that stores minimal vertical resolution. Should be set to 480
MIN_HORIZONTAL_RESOLUTION	Constant that stores minimal horizontal resolution. Should be set to 640.
WAIT_RESOLUTION	Wait interval for resolution test. Should be set to 5 seconds.
WAIT_COLOR	Wait interval for color test. Should be set to 5 seconds.
WAIT_FOCUS	Wait interval for focus test. Should be set to 5 seconds.
WAIT_GEOMETRY	Wait interval for geometry test. Should be set to 5 seconds.

Table 3. Monitor variables, constants and structures

Determining the supported screen resolutions and modes

1. Determine all the supported resolutions using EnumDisplaySettings( ) API function.

2. Only get resolution greater than MIN_VERTICAL_RESOLUTION and MIN_HORIZONTAL_RESOLUTION. Only get resolutions for the current refresh rate; do not use different refresh rates settings.
3. If there was a problem getting resolution settings, display warning info message WARNING_RESOLUTION_SETTINGS and do not run resolution test.

Determining the supported screen resolutions and modes

1. Display message box MSG_INFO_RESOLUTION
2. Display info message MSG_INFO_RES_TEST
3. For each supported resolution
    a. Display info message MSG_INFO_CHANGE <mode>
    b. Attempt to change current resolution to selected mode.
    c. If the resolution cannot be changed or it requires a reboot, generate warning WARNING_RESOLUTION_CHANGE <mode> and try go to the next resolution test.
    d. Wait WAIT_RESOLUTION seconds.
    e. Restore the default user resolution
    f. If there was a problem restoring the current resolution, generate error ERR_RESTORE_RESOLUTION
    g. Prompts the user with message PROMPT_RESOLUTION
    h. If the user answers "No", then generate error ERROR_RESOLUTION <mode>

For the following tests, if there is any problem when displaying a full-screen window, then display warning message WARNING_SCREEN_SIZE. Also, for the following tests, if there is any problem painting the window, display the warning message WARNING_SCREEN_PAINT.

Color test

1. Display info message MSG_INFO_COLOR_TEST
2. Display message box MSG_INFO_COLOR
3. For colors = {Black, White, Red, Green, Blue} do
    a. Display a full-screen with the selected color
    b. Wait WAIT_COLOR seconds
    c. Prompt the user with message PROMPT_COLOR <color>
    d. If the user answers "No", then generate error ERROR_COLOR <color>

Focus test

1. Display info message MSG_INFO_FOCUS_TEST
2. Display message box MSG_INFO_FOCUS
3. Display a full-screen with the focus test (small characters in the screen)
4. Wait WAIT_FOCUS seconds
5. Prompt the user with message PROMPT_FOCUS
6. If the user answers "No", then generate error ERROR_FOCUS

Geometry test

1. Display info message MSG_INFO_GEOMETRY_TEST
2. Display message box MSG_INFO_GEOMETRY
3. Display a full screen with the geometry test.
4. If there is any problem painting lines on the screen, display info warning WARNING_GEOMETRY_LINES
5. Wait WAIT_GEOMETRY seconds
6. Prompt the user with message PROMPT_GEOMETRY
7. If the user answers “No”, then generate error ERROR_GEOMETRY

PCI bus test

Implementation aspects

The PCI bus diagnostic shall have five phases. The first phase shall be responsible for enumerating all PCI devices. The second phase shall verify information in the command register. Then, the diagnostic shall test information in the status register. Next, the information in the Built-In Self Test (BIST) register shall be tested. Finally, the value of the interrupt line register shall be validated.

- First phase: It shall enumerate all PCI devices, getting PCI common header information and detecting the device type.
- Second phase: It shall verify information in the command register to detect if the device is enabled or disabled and if parity error is enabled or disabled.
- Third phase: It shall verify according to status register information if there is a parity or system error.
- Fourth phase: It shall verify if devices support BIST, and if so, it shall verify if the test has passed.

Action flow

Variables, constants and structures

Name	Description
MAX_PCI_BUS_NT	Constant that indicates the maximum number of busses. It should be set to 32.
MAX_PCI_SLOTS	Constant that indicates the maximum number of slots. It should be set to 256.
MAX_PCI_DEVICES	Constant that indicates the maximum number of devices. It should be set to 32.
MAX_PCI_FUNCTIONS	Constant that indicates the maximum number of functions for multifunction devices. It should be set to 8.
INVALID_VENDOR_ID	Any VendorID greater than this constant is invalid. It should be set to 0x0FFFF.

Table 4. PCI bus variables, constants and structures

Enumerating PCI devices

1. Display MSG_PCI_ENUMERATION
2. First, the OS must be detected. If OS is Win98 or WinMe for each possible combination of MAX_PCI_BUS, MAX_PCI_DEVICES, MAX_PCI_FUNCTIONS, try to get the information (value of PCI common header registers) from the device using 0xCF8 0xCFC I/O ports. Otherwise, for each possible combination of MAX_PCI_BUS, MAX_PCI_SLOTS, try to get the information from the device using the DeviceIOControl( ) function to interact with HPDOM's device driver.
3. For each combination in item 1, the VendorID value must be specified. If it is greater than INVALID_VENDOR_ID, go to the next combination and ignore the information; else, store the information from the device in an instance of CPciDevice class, add it to a dynamic list (CPciDeviceList), and show MSG_PCI_DEVICE_DETECTED to user.
4. If in WinNT systems the diagnostic is unable to communicate with pciinfo driver the ERROR_PCI_DRIVER must be generated; else, after all possible combinations, no device was

found or if any other error occurs in the enumeration phase, the diagnostic must generate ERROR_PCI_READ_REGISTERS error.

Display MSG_PCI_REGISTERS_TEST

Command register test

1. For each device in CpciDeviceList, read the corresponding Command register value.
2. Verify the first two bits (0, 1) of this register. If both bits are reset, generate a WARNING_PCI_DEVICE_DISABLED warning.

Status register test

1. For each device in CpciDeviceList, read the corresponding status register value.
2. If bit 14 is set, generate an ERROR_PCI_DEVICE_SYSTEM error.
3. If bit 15 is set:
   a. If bit 6 in command register is set, generate the ERROR_PCI_DEVICE_PARITY error.
   b. If bit 6 in command register is reset, generate the WARNING_PCI_DEVICE_PARITY warning.

BIST register test

1. For each device in CpciDeviceList, read the corresponding BIST register value.
2. If bit number 7 is set, it indicates that the device supports BIST.
3. If the device supports BIST, verify its error code reads the first four bits (0, 1, 2 and 3). These three bits must be reset; otherwise the BIST detected an error and error ERROR_PCI_DEVICE_BIST must be generated.

Parallel port test

Implementation aspects

This diagnostic shall have two phases. The first phase shall be responsible for locating the corresponding LPT ports associated to the machine's parallel ports. This part searches the Windows registry and uses basic API calls in order to accomplish this functionality. The second part of the test shall be responsible for sending a couple of commands to the port, changing basic configuration functions and settings, but without sending read or write commands to the port.

- First phase: This phase shall be responsible for locating the corresponding LPT ports associated to the machine's parallel port. This part searches the Windows registry (in window 98) and uses API calls (Windows NT family) in order to accomplish this functionality. The absence of the referred information in Windows registry shall indicate an error condition since no parallel ports are detected.

- Second phase: This phase shall be responsible for issuing commands to the parallel port. These shall be read/write operations to the parallel port accumulator. There is no test for printing functionality since this is handled by the OS and it can be redirected not only to a parallel port, but also to the network or other type of devices.

Action flow

Detecting parallel port "LPT" (1st phase)

1. If OS = Windows NT, 2000 or XP
    a. For I = 1 to 20 do
        i. Use QuerDosDevices() with "LPT<I>" to detect if the device exists
    b. If no parallel ports are detected set error to ERROR_PARALLEL_ NOTFOUND
2. Else if OS = Windows 98 or Millennium
    a. For each key HKEY_LOCAL_MACHINE\Enum\Acpi\<X>\<Y>, if value PortName contains substring "LPT", then store the value of PortName in the list of parallel ports.
3. If no parallel ports are detected, set error to ERROR_PARALLEL_NOTFOUND

Sending commands to the parallel port (2nd phase)

1. Use CreateFile( ) to open each parallel port szPortName concatenated with ":". Use option OPEN_EXISTING for dwCreationDisposition and 0 for dwFlagsAndAttributes.
    a. In case of error, set error to ERROR_DEVICE_OPEN
2. Use API function GetCommState() to get the parallel port state
    a. In case of error, set error to ERROR_DEVICE_GETSTATE
3. Close port handles

Serial port test

Implementation aspects

The serial port diagnostic shall have two phases. The first phase shall be responsible for locating the corresponding COM ports associated to the machine's serial port. This part searches the Windows registry in order to accomplish this functionality. The second part of the test shall be responsible for sending a couple of commands to the port and changing basic configuration functions and settings. Read and write commands shall not be sent to the port.

- First phase: This phase shall be responsible for locating the corresponding COM ports associated to the machine's serial port. This part searches the Windows registry in order to accomplish this functionality. It shall take into account the current Operating System, since the location of the COM devices may vary according to it. The absence of the referred information in Windows registry shall indicate serial ports are not detected and generate an error condition.

- Second phase: This phase shall be responsible for issuing commands to the serial port. This command shall change the current baud rate, flow control and error detection settings. If any of these commands cannot be completed, an error shall be generated.

Action Flow

Detecting serial port "COM" (1st phase)

1. Search registry key HKEY_LOCAL_MACHINE\Hardware\DeviceMap\SerialCom
   a. If the key cannot be found, set error ERROR_SERIAL_REGNOTFOUND
2. Under this same key, list all values. The contents of these values are the names of each serial port device. Example: COM1, COM2.
   a. If no value is found, set error to ERROR_SERIAL_NOTFOUND

Sending commands to the serial port (2nd phase)

1. Use CreateFile( ) to open the serial port szPortName concatenated with ":". Use option OPEN_EXISTING for dwCreationDisposition and 0 for dwFlagsAndAttributes.
   a. In case of error, set error to ERROR_DEVICE_OPEN
2. Use API function GetCommState() to get the serial port state
   a. In case of error, set error to ERROR_DEVICE_GETSTATE
3. Varying baud rates [BaudRate=9600, 14400, 19200, 28800, 57600, 115200] do
   a. Clear all characters in the communication buffer using PurgeComm function
      i. In case of error, set error code to ERROR_DEVICE_PURGECOMM
   b. Set the serial port baud rate to BaudRate using SetCommState and the configuration obtained with original GetCommState()
      i. In case of error set error to ERROR_DEVICE_SETCOMM<BaudRate>
   c. Using SetCommState, set RTS/CTS error control method
      i. In case of error, set error code to ERROR_DEVICE_RTSCTS
4. Close port handles

Video test

Implementation aspects

The video diagnostic shall have five phases. The first phase shall be responsible for enumerating all PCI devices and checking information in the PCI registers. The diagnostic shall then set different resolutions to test video modes. Next, the diagnostic shall write text in the video memory. When this is completed, the diagnostic shall fill the video buffer with some color patterns. Finally, it shall test data transfer from/to video memory.

- First phase: It shall enumerate all PCI devices, getting PCI common header information and detecting the device type. The command, status, BIST and interrupt line PCI registers information must be checked when the device is a video adapter.
- Second phase: This phase shall search the registry and Windows API looking for supported resolutions. Then, diagnostics shall attempt to change to each supported resolution in a valid refresh rate configuration. This test shall be similar to the resolution test performed by the monitor diagnostic.
- Third phase: It shall print text in random colors and sizes on the screen using video memory.
- Fourth phase: It shall fill the video frame buffer with random color patterns.
- Fifth phase: This phase shall test video memory. It shall be written and read in order to check if data was written correctly.

Algorithm

Variables, constants and structures

Name	Description
MAX_PCI_BUS	Constant that indicates the maximum number of PCI busses. It should be set to 256.
MAX_PCI_SLOTS	Constant that indicates the maximum number of PCI slots. It should be set to 256.
MAX_PCI_DEVICES	Constant that indicates the maximum number of PCI devices. It should be set to 32.
MAX_PCI_FUNCTIONS	Constant that indicates the maximum number of PCI functions for multifunction devices. It should be set to 8.
INVALID_VENDOR_ID	Any PCI VendorID greater than this constant is invalid. It should be set to 0x0FFFF.
MIN_VERTICAL_RESOLUTION	Minimal vertical resolution. Should be set to 480.
MIN_HORIZONTAL_RESOLUTION	Minimal horizontal resolution. Should be set to 640.
WAIT_RESOLUTION	Wait interval for resolution test. Should be set to 1-5 seconds.
SCREEN_WIDTH	Width of screen used in DirectX tests. It must be set to 640.
SCREEN_HEIGHT	Height of screen used in DirectX tests. It must be set to 480.
SCREEN_BPP	Bits Per Pixel used in DirectX tests. It must be set to 16.

Table 5. Video variables, constants and structures

Enumerating PCI devices and testing PCI registers

1. Display message MSG_PCI_REGISTERS_TEST
2. The OS must be detected first. If OS is Win98 or WinMe, then for each possible combination of MAX_PCI_BUS, MAX_PCI_DEVICES, MAX_PCI_FUNCTIONS get the information (value of PCI common header registers) from the device using 0xCF8 0xCFC I/O ports. Otherwise, for each possible combination of MAX_PCI_BUS, MAX_PCI_SLOTS, get the information from the device using the DeviceIOControl( ) function to interact with HPDOM's device driver.
3. For each combination above, the VendorID value must be checked. If it is greater than INVALID_VENDOR_ID, go to the next combination and ignore the information; else store the information from the device in an instance of CPciDevice class and add it to a dynamic list (CPciDeviceList).
4. If in WinNT the systems diagnostics are unable to communicate with pciinfo driver, the ERROR_PCI_DRIVER must be generated; else, after all possible combinations, no device was found or if any other error occurs in the enumeration phase, the diagnostic must generate ERROR_PCI_READ_REGISTERS error.

Command register test

1. For each video adapter device in CPciDeviceList, read the Command register value.
2. Verify the first two bits (0, 1) of this register. If both bits are reset, generate a WARNING_PCI_DEVICE_DISABLED warning.

Interruption line test

1. For each video adapter device in CPciDeviceList, read the Interruption Line register value.

Status register test

1. For each video adapter device in CPciDeviceList, read the Status register value.
2. If bit 14 is set, generate an ERROR_PCI_DEVICE_SYSTEM error.
3. If bit 15 is set:
   a. If bit 6 in command register is set generate the ERROR_PCI_DEVICE_PARITY error.
   b. If bit 6 in command register is reset, generate the WARNING_PCI_DEVICE_PARITY warning.

BIST register test

1. For each video adapter device in CPciDeviceList, read the BIST register value.
2. If bit *7* is set, it indicates that the device supports BIST.
3. If the device supports BIST, verify its error code reading the first three bits (0, 1, 2 and 3).
4. These three bits must be reset, otherwise the BIST detected an error and error ERROR_PCI_DEVICE_BIST must be generated.

Determining and testing the supported screen resolutions and modes

1. Determine and test the supported screen resolutions and modes.
2. Determine all the supported resolutions using EnumDisplaySettings( ) API function.
3. Only get resolutions greater than MIN_VERTICAL_RESOLUTION and MIN_HORIZONTAL_RESOLUTION. Only get resolutions for the current refresh rate; do not use different refresh rates settings.
4. If there was a problem getting resolution settings, generate error ERROR_RESOLUTION_SETTINGS and do not run the resolution test.

5. Display info message MSG_RESOLUTION_TEST.
6. For each supported resolution, do the steps below.
    a. Display info message MSG_CHANGE_RESOLUTION <mode>
    b. Attempt to change current resolution to selected mode using ChangeDisplaySettings( ) functions.
    c. If the resolution cannot be changed, the diagnostic generates error ERROR_RESOLUTION_CHANGE <mode> and try going to the next resolution test. If a reboot is required to change the resolution, a warning WARNING_RESOLUTION_CHANGE must be generated.
    d. Wait WAIT_RESOLUTION seconds.
    e. Restore original resolution. If this fails, the error ERROR_RESOLUTION_RESTORE must be generated.

Remarks: This test may seem like the resolution test for monitor diagnostic, but it does not prompt any questions to the user.

Display MSG_VIDEO_MEMORY_TEST

Painting color patterns in video frame buffer

1. First, the OS must be detected. If OS is WinNT, it must have SP3 and DirectX installed. If OS is WinNT and it doesn't have SP3 and DirectX installed, show WARNING_NT_OS and do not run the test.
2. Create a DirectDraw object.
3. Set cooperative level to exclusive and full screen.
4. Use SetMode( ) to set SCREEN_WIDTH x SCREEN_HEIGHT x SCREEN_BPP resolution.
5. Create a surface using capabilities flags DDS_CAPS_PRIMARYSURFACE, DDS_CAPS_VIDEOMEMORY and DDS_CAPS_LOCALVIDMEM (only if OS isn't WinNT).
6. If any one of the steps above fails, the error WARNING_DIRECTX_FAILED must be generated.
7. Use Blt( ) surface method to paint 15 random color patterns on screen. The error ERROR_FRAME_BUFFER_COLOR must be generated if Blt( ) method failed.

Testing data transfer in video memory

1. First, the OS must be detected. If OS is WinNT it must have SP3 and DirectX installed. If OS is WinNT and it doesn't have SP3 and DirectX installed show WARNING_NT_OS and do not run the test.
2. Create a DirectDraw object.
3. Set cooperative level to exclusive and full screen.
4. Use SetMode( ) to set SCREEN_WIDTH x SCREEN_HEIGHT x SCREEN_BPP resolution.
5. Create a surface using the capabilities flags DDS_CAPS_PRIMARYSURFACE, DDS_CAPS_VIDEOMEMORY and DDS_CAPS_LOCALVIDMEM (only if OS isn't WinNT).
6. If any one of the steps above fails, the error WARNING_DIRECTX_FAILED must be generated.
7. Use GetPixelFormat( ) to get RGB color masks.
8. For each screen pixel, choose a random color and random RGB mask (or RGB masks combination), use Lock( ) surface method to get a pointer to video memory, write pixel with chosen color and mask. Then, read pixel from video memory and compare it with written

value. If it does not remain the same, the error ERROR_VIDEO_MEMORY_TRANSFER must be generated. Finally, use Unlock( ) method to yield video memory. The warning WARNING_LOCK_SURFACE must be generated if the Lock( ) method fails. Do these steps five times.

Writing texts in video memory

1. First, the OS must be detected. If OS is WinNT it must have SP3 and DirectX installed. If OS is WinNT and it doesn't have SP3 and DirectX installed show WARNING_NT_OS and do not run the test.
2. Create a DirectDraw object.
3. Set cooperative level to exclusive and full screen.
4. Use SetMode( ) to set SCREEN_WIDTH x SCREEN_HEIGHT x SCREEN_BPP resolution.
5. Create a surface using capabilities flags DDS_CAPS_PRIMARYSURFACE, DDS_CAPS_VIDEOMEMORY and DDS_CAPS_LOCALVIDMEM (only if OS isn't WinNT).
6. If anyone of the steps above fails, the error WARNING_DIRECTX_FAILED must be generated.
7. Uses GetDC( ) to get Device Context of DirectDraw object.
8. Write some text strings in random sizes and colors using Win32 GDI functions. The error ERROR_VIDEO_MEMORY_TEXT must be generated if any errors occur when writing text.
9. Display message MSG_DIAGNOSTIC_FINISHED.

Floppy drive test

General technical overview

The Win32 Online HPDOM Floppy Drive Diagnostics provide the capability to diagnose the floppy drive without rebooting the system. This diagnostic is not an alternative to the existing 16-bits diagnostics. The following sections and subsections explain precisely how this diagnostic works. In order to minimize risks, this diagnostic is not a driver based diagnostic (Kernel level) but an application diagnostic.

All algorithms have been designed in order to have only one multi-OS floppy diagnostic for:

1. Win98 SE
2. Windows NT4.0 Workstation, Server (with at least SP4)
3. Windows 2000 Professional, Advanced Server and XP

Diagnostic principle

One of the main concerns of this diagnostic is to test the physical drive without corrupting the data on the media; therefore, this diagnostic does not write to the media.

Main algorithm

Here is the main algorithm of the floppy drive diagnostic.

1. Check if the drive type is supported.
2. Calculate sector number.
3. Flush floppy cache.
4. Stress the floppy head to detect a mechanical problem (head failure).
5. Read all sector in respect of the media size.

Floppy drive tests algorithm

For NT4, Windows 2000, XP, the floppy diagnostic moves the head and reads the sector through the windows API. In Windows 98, the diagnostic calls the interruptions 0x21 and 0x25 to lock the floppy drive, move the head and read sectors.

Check the drive type

The floppy diagnostic tests the media size of the floppy drive:

1. 720K
2. 1.44MB
3. 2.88MB

For other formats, the diagnostic returns an error: ERR_MEDIA_TESTED

Flush the floppy drive cache

For Windows 98, it is necessary to flush the floppy cache before directly testing the floppy drive. The operating system reads from cache instead of directly from the media and doesn't allow the drive head to move, without this action. Flushing the cache also includes locking and unlocking the floppy drive at different levels.

Funnel read test

In order to detect a mechanical problem with the drive head, the diagnostic stresses the floppy drive head. The test moves the drive head to each extremity of a funnel as the diameter decreases.

The floppy drive reads a sector following this algorithm:

```
While (I< diskSize){
        I=+ diskSize/20;
        While (k <2) {
                sectorToRead = (1-k)*i + k*(diskSize - i);
        }
}
```

Linear read test

In order to detect that the head is well situated and not damaged, the diagnostic reads linearly all sectors on the media.

Limitations

The floppy diagnostic does not test the write operation in order to not corrupt data on the media.

Error Codes

Error Code	Number	Description
ERR_LOCK_VOLUME	0x0501	Cannot lock the floppy.
ERR_GET_DISK_GEOMETRY	0x0502	Cannot get the geometry of the media.
ERR_READ_DATA	0x0503	Cannot read data.
ERR_MOVE_HEAD	0x0504	Cannot move the drive head of the floppy.
ERR_DETECT_MEDIA	0x0505	Cannot detect a media in the floppy drive.
ERR_MEDIA_TESTED	0x0506	Device not supported.
ERR_READ_SECTOR	0x0507	Cannot access or read the sector.
ERR_UNLOCK_VOLUME	0x0508	Cannot lock or unlock the floppy.
ERR_FLOPPY_NOT_FOUND	0x0509	The floppy name specified is not found.
ERR_FLOPPY_OPEN_DRIVE	0x050A	Unable to open the floppy drive.

Table 6. Floppy error codes

CDROM drive test

General technical overview

The Win32 Online HP DOM CDROM Diagnostic has the capability to diagnose the CDROM drive without rebooting the system. This diagnostic is not an alternative to the existing 16-bits diagnostic. The following sections and subsections explain precisely how this diagnostic works. In order to minimize risks, this diagnostic is not a driver based diagnostics (Kernel level) but an application diagnostics.

All algorithms have been designed in order to have only one multi-OS CDROM diagnostic for:

1. Win98 SE
2. Windows NT4.0 Workstation and Server (with at least SP4)
3. Windows 2000 Professional, Advanced Server and XP

The diagnostic time depends on the media size.

Diagnostic principle

Main algorithm

Here is the main algorithm of the CDROM diagnostic.

1. Check the presence and the type of media.
2. Calculate number of sectors.
3. Flush the CDROM cache (for Windows 98).
4. Move head linearly.
5. Move head to stress CD-ROM head in order to detect a mechanical problem.
6. Read all sectors following the media size and the quick/normal mode.
7. Test open/close operation.

CDROM tests algorithm

For NT4, Windows 2000 and XP, the CDROM diagnostic moves the drive head and reads the sector through the windows API. In Windows 98, the diagnostic runs a 16-bit process that calls the "mscdex" functions.

Check the drive type

The CDROM diagnostic cannot run if the media inserted is not an audio or blank recordable CD. In this case, the diagnostic gives the following error: ERR_CDROM_FORMAT_NOT_VALID.

Flush the CDROM drive cache

For Windows 98, it is necessary to flush the floppy cache before directly testing the CDROM drive. The operating system reads from cache and not directly from the media and doesn't allow the drive head to be moved, without this action. This action needs to lock and unlock the floppy drive at different levels.

Funnel read test

In order to detect a mechanical problem with the CDROM drive head, the diagnostic stresses it. The test moves the drive head to each funnel extremity while the diameter decreases.

The CDROM drive read a sector following this algorithm:

```
While (I< diskSize){
        I=+ diskSize/20;
        While (k <2) {
                sectorToRead = (1-k)*i + k*(diskSize - i);
        }
}
```

Linear read test

In order to detect that the drive head is well positioned and not damaged, the diagnostic reads linearly all sectors on the media. This test runs only in normal mode.

Open/close operation

This test checks the drive's ability to open and close the door. Before running this test, the diagnostic checks if the door is closing automatically. If not, this test is not running.

Limitations

The diagnostic tests the CDROM drive through a media. It is possible that the diagnostic returns an error, which comes from the media and not from the drive itself. In this case, it s recommended to run again the diagnostic with another media. For certain media (with an auto run), a process starts when the media is inserted. In this case and in Windows 98, it is difficult to run all diagnostic sub -tests to fully diagnose the CD-ROM drive.

Error Codes

Error Code	Number	Description
ERR_CDROM_FORMAT_NOT_VALID	0x0401	CDROM format not valid
ERR_OPEN_CDROM_DRIVE	0x0402	Unable to access the device
ERR_INVALID_CDROM	0x0403	Device is present but does not answer from mscdex reques
ERR_RUN_PROCESS	0x0404	Diagnostic is not correctly installed
ERR_EJECT_MEDIA	0x0405	Unable to open the door
ERR_CDROM_SIZE_NOT_VALID	0x0406	CDROM disk size is not valid
ERR_MEDIA_NOT_FOUND	0x0407	CDROM disk not found
ERR_IDENTIFY_DRIVE	0x0408	Unable to find specified device
ERR_READ_MEDIA_GEOMETRY	0x0409	Unable to evaluate the geometry of the CDROM disk
ERR_CDROM_READ_SECTOR	0X040A	Unable to read a sector - user must test again with another media
ERR_NO_CDROM_FOUND	0x040B	Unable to find CDROM drive

Table 7. CDROM error codes

Hard disk drive test

Implementation aspects

For the hard disk drive (HDD) diagnostic, we are using the "SMART" tests embedded in the HDD itself. If the HDD doesn't support SMART, we use a generic test.

Diagnostic main algorithm

For Windows NT4.0 and 2000/XP:

1. If the HDD is SMART compliant, we extract information about it:
   - Model
   - Capacity
   - Manufacturer
   - Serial Number
   - Number of: cylinders, heads, sectors and sectors per track
   - Location
   - The different partitions found
   - The total space and the free space

   Offline tests (either quick or normal mode) are run once the above information is obtained.

   After we check for the standard attributes, check that the attribute value is not less than the threshold value specified by the manufacturer.
2. Else we run the generic test.
3. The log file shall store information about the disk and the results of the test.

For Windows 98:

1. During the installation of HPDOM, the driver smartvsd.vxd is copied onto the directory "windows\system\iosubsys" if no CD-writer is installed. This driver must be present to run the smart test.

The remainder of the diagnostic is the same as NT4.0 (above).

SMART diagnostic test

Acronym for Self-Monitoring, Analysis and Reporting Technology (SMART), an open standard for developing disk drives and software systems that automatically monitors a disk drive's health and reports potential problems. Ideally, this should allow you to take proactive actions to prevent impending disk crashes.

The SMART system for disk drives is designed to revolutionize overall system and data reliability. The system is comprised of software that resides both on the disk drive and on the host computer. The disk drive software monitors the internal performance of the motors, media, heads, and electronics of the drive, while the host software monitors the overall reliability status of the drive. The reliability status is determined through the analysis of the drive's internal performance level and the comparison of internal performance levels to predetermined threshold limits.

The SMART system drives extend this technology by designing the diagnostics into the drive. There, the diagnostic routines can be more precise because they are designed for a specific drive design. They can also be more effective because they have access to the internal performance and calibration measurements collected by the drive's controller.

SMART drives can be automatically monitored for impending failure conditions. Drive manufacturers can use the specific internal diagnostics best suited to their drive. SMART can detect and report

failure conditions that originate in the field due to shock, vibration, temperature, and voltage extremes.

Not all hard drive failures are predictable, but analyzing the mechanical attributes of the drive can predict failures.

Generic test

Once the geometry of the disk and the last piece of log information is obtained through the registry, a funnel seek is made.

In quick mode, the first and the last mega octet on the disk is read.

In normal mode, the entire disk is read.

Standard attributes

The SMART attribute is a specific property of disk being monitored. Every SMART attribute has a set of properties: attribute value, its threshold, worst attribute value and raw value. Specific threshold is assigned to each attribute. If one or more attribute values is less than or equal to the corresponding attribute thresholds, then the device reliability status indicates an impending degrading or fault condition.

Note: Some attributes are considered life-critical and others are merely "informative".

Attribute value

Attribute values are used to represent the relative reliability of individual performance or calibration attributes. The current attribute value is the normalized raw attribute data. The value varies between 1 and 100. The closer the value gets to 1, the higher the possibility of a failure. The device compares the attribute values with thresholds. When the attribute values are larger than the thresholds, the device is operating normally.

Raw attribute value

This is a raw attribute data. Usually it shows the exact amount of time, attempts or errors. For example: the raw value of attribute Temperature is a drive temperature in Celsius degrees, the raw value of Power on hours count attribute is a amount of hours when drive was in power-on state. You can view the raw attribute value in the Text Report (Open SMART Info tab and press Report button).

Attribute threshold

This is the lowest limit of a varying attribute value. SMART compares the attribute values with the thresholds to identify a failure. Each attribute value has a corresponding attribute threshold limit. The device manufacturer through design and reliability testing and analysis determines the attribute threshold numerical value. Attribute thresholds are set at the device manufacturer's factory and cannot be changed in the field. The valid range for attribute thresholds is from 1 through 253 decimal.

Worst value

This is the worst attribute value among the attribute values collected to date. This value indicates the state nearest to a failure so far.

SMART attribute meaning

Each SMART attribute has its own value, meaning, and importance. Attributes are vendor and drive specific, but some of them are common for all manufacturers:

Attribute	Description
Power On Hours Count	This attribute shows drive aging; i.e., how long the drive is powered on. The raw value of this attribute shows amount of hours of the power - on state.
Start/Stop Count	Amount of drive's start/stop cycles.
Reallocated Sector Count	Indicates amount of re-mapped sectors. If the sector becomes bad for some reason, then the drive replaces it with a spare sector from a special spare area. The value of 100 means that no sectors were re-mapped. The raw value of this attribute shows exact amount of reallocated sectors.
Spin Up Time	The raw value of this attribute indicates average time to spin up the drive spindle.
Spin Retry Count	Count of retry of drive spindle spin start attempts. The raw value indicates amount of retries.
Power Cycle Count	Count of disk power cycles. The raw value indicates amount of power cycles.
Raw Read Error Rate	This value depends on read errors, and disk surface condition and indicates a read retry rate. Lower values indicate that there is a problem with either disk surface or read/write heads.
Write Error Count	The raw value indicates amount of write errors.
Seek Error Rate	Average rate of seek errors. This attribute indicates a state of head positioning mechanism. Lower values show that there is a problem with head positioning.
Seek Time Performance	Disk seek system performance.
Temperature	If the drive contains a dedicated thermal sensor, then this attribute shows a drive temperature. Active SMART supports drives with temperature sensors.
Throughput Performance	Overall throughput performance of the drive.
Off-line Scan Uncorrectable Sector Count	Amount of error sectors detected during the last off-line scan.
Current Pending Sector Count	Amount of pending sectors. If the sector issues an error during read or write, the drive marks it as pending for a certain time before replacing this error sector with a spare one.

Table 8. SMART attributes

Errors and warnings

Error Code	Number	Description
ERR_IDE_GT_ACCESS_DRIVE	0x0A01	Unable to access the hard disk.
ERR_IDE_GT_READ_GEOMETRY_DRIVE	0x0A02	Unable to evaluate the geometry of the hard disk.
ERR_IDE_GT_READ_DATA	0x0A03	Unable to read data.
ERR_IDE_GT_MOVE_HEAD	0x0A04	Unable to move the head.
ERR_IDE_INSTALL_DIAGNOSTIC	0x0A05	The diagnostic is not properly installed.

ERR_IDE_DISK_GEOMETRY	0x0A06	Unable to evaluate the geometry of the hard disk.
ERR_IDE_SMARTDRV_NOT_FOUND	0x0A07	Unable to find the SMART driver.
ERR_IDE_READ_DATA	0x0A08	A read data error occurred.
ERR_IDE_WRITE_DATA	0x0A09	A write data error occurred.
ERR_IDE_UNKNOW_FAILURE	0x0A0A	An unknown error occurred.
ERR_IDE_EXECUTE_SELF_TEST	0x0A0B	Unable to execute the offline test.
ERR_IDE_SERVO_FAILURE	0x0A0C	A server failure occurs.
ERR_IDE_HDD_NOT_IDENTIFIED	0x0A0D	Unable to find the specified hard disk.
ERR_GENERIC_TEST	0x0A0E	Unable to find the SMART driver.
ERR_IDE_ OFFLINE_TEST_RUN	0x0A0F	The offline test does not run correctly.
ERR_SMART_ERROR	0x0A10	One attribute value is less than or equal to its corresponding attribute threshold.

Table 9. Hard disk drive errors and warnings

Keyboard test

Main failures

Main failures encountered are:

1. Interrupt conflict
2. Address conflict
3. Key multimedia programmed badly
4. Multimedia driver version not correct or absent
5. Driver absent or damaged
6. Connectivity problem

Failure detection

1. Check the configuration of the multimedia installation keyboard
2. Check each keypad
3. Verify the connectivity
4. Verify the BIOS version for the USB keyboard
5. Check LED's

General overview

A keyboard diagnostic is defined and represented by:

1. A bitmap displayed in the diagnostic dialog box,
2. An XML file containing some specific information on some HP keyboards, and
3. A text file containing all key coordinates on the bitmap.

Each keyboard has a specific associated bitmap. The following list gives different keyboards supported by this diagnostic:

1. KING Keyboard (Multimedia)
2. KROKE Keyboard (Multimedia)
3. Standard Keyboard
4. USB Keyboard (Multimedia)

LAN test

General technical overview

The following sections and subsections explain precisely how the Local Area Network (LAN) diagnostic works. This diagnostic, which is a driver based diagnostic, is supported on the following Microsoft Operating Systems:

1. Windows 98 SE
2. Windows NT4.0 Workstation (with at least SP4.0)
3. Windows 2000 Professional and XP

Aim of diagnostic

This diagnostic is a generic LAN diagnostic and provides the capability to catch network activity (it captures network packets like a network spy). LAN diagnostic is independent of network protocol and also allows quick isolation if the problem is linked to a software configuration or linked to a specific hardware problem. If it is a hardware problem, a vendor diagnostic shall be proposed to the user in order to diagnose the specific device more deeply.

Diagnostic main algorithm

Here is the main sequence algorithm flow chart:

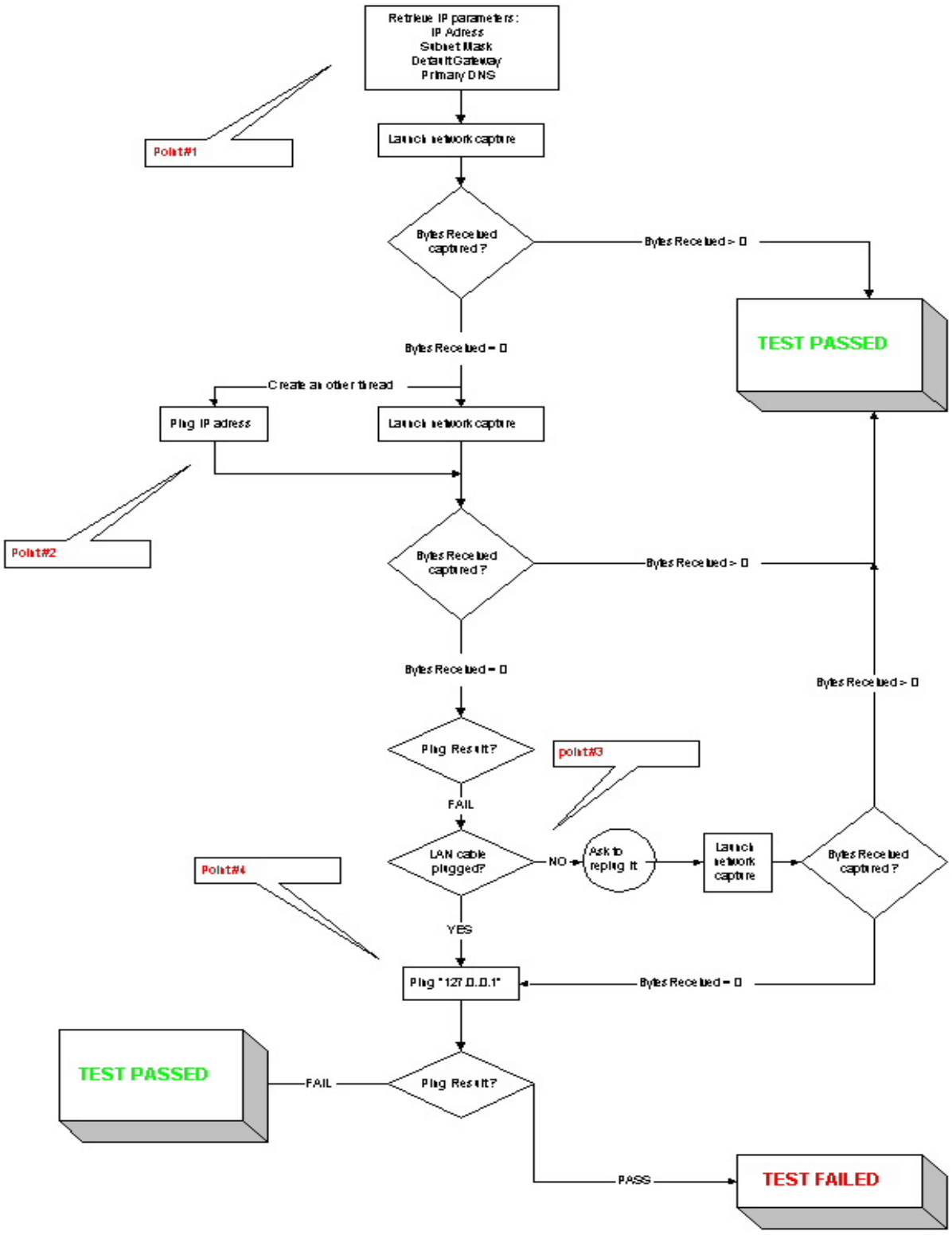

Figure 3. LAN main algorithm

Remarks

Point #1

Methods used to retrieve these parameters are OS dependant. For Windows 98 SE and Windows 2000, all IP configuration parameters are retrieved thanks to the IP Helper API functions and Data

Structures. But for Windows NT4.0, most of the functions and structures coming from the API are not supported. Consequently all parameters are directly retrieved inside the registry.

Point #2

At this point three possibilities may exist:

- The LAN adapter has a hardware problem
- There is no network activity
- The LAN cable is unplugged

Consequently, in order to be sure it is not linked to a lack of network activity, 200 pings at the system IP address are done in parallel of the capture. This creates some traffic and allows at the same time to have some ICMP statistics.

Point #3

LAN Cable detection: If the LAN cable is unplugged (i.e., number of sent and received ICMP packets is null), the user is asked to re-plug it and capture is re-launched.

Point #4

This step is done in order to check if the problem is not linked to a bad configuration of the TCP/IP stack. In pinging 127.0.0.1, we can verify the internal loop back and validate TCP/IP stack.

Packet driver package

Overview

HP DOM LAN diagnostic is a driver based diagnostic. The capability to catch network packets is provided by a specific driver called:

- WinPcap: the Free Packet Capture Architecture for Windows.

Furthermore, a specific library called "Packet Capture library" is also delivered and it provides a high level interface to packet capture systems. All packets on the network, even those destined for other hosts, are accessible through this mechanism.

Install and uninstall procedures

Before using the LAN diagnostic, we must install WinPcap driver and DLLs. This is done by an installation applet, which automatically detects the operating system and installs the correct components.

Remarks

1. It would be better to install driver and DLLs at the same time as HPDiags module. We must include in the installation batch file a command line allowing launch of this auto-installer.

2. If driver and DLLs are not installed on the system, HP DOM LAN diagnostic registration fails.

    - To remove winpcap from the system, go to the control-panel, click on "add/remove programs" and then select "WinPcap".

LAN cable connection

One of the most frequent problems is usually the LAN cable. Most of the time, the user doesn't verify if the LAN cable is plugged in well. This cable detection method is OS independent.

Algorithm

Win32 API provides some IP Helper structures and functions allowing ICMP statistics. Thanks to them, we can have a simple way to detect if there is a LAN cable problem.

1. Initialize MIB_ICMP ICMP structure
2. Get OUT/IN coming ICMP messages with GetIcmpStatistics( ) function
3. Ping local system IP address
4. Get OUT/IN coming ICMP Messages
5. Extract only OUT/IN number of echo requests received or sent.
6. IF IN and OUT number echo requests = 0, THEN Network Cable is unplugged or there is a network connection problem.

Remarks

- Only dwEchos of MIBICMPSTATS field is used
- In fact, this algorithm detects if the network adapter is disconnected from the network. Sometimes the problem can also come from the LAN cable itself or can be unplugged on the other cable side.
- Windows 2000 already notifies when the LAN cable is unplugged. It is also possible to detect a connection problem in trapping the two following events.

Event Source	Event Type	EventID (Hexa)	Description
TCPIP	EVENTLOG_INFORMATION_TYPE	0x0000106A	LAN Cable is unplugged.
TCPIP	EVENTLOG_INFORMATION_TYPE	0x00001069	LAN Cable has been re-plugged.

Table 10. LAN cable Windows 2000 events

Information displayed for IP configuration

In order to help the user to troubleshoot an IP configuration problem, this diagnostic displays the following information concerning the current network configuration:

1. IP address
2. Subnet Mask
3. Default Gateway
4. Primary DNS

Remarks

If IP configuration parameters are not available on the system, the string "Not Available" is displayed. The following subsection gives implementation information to retrieve IP configuration under Windows 9x, 2000 and Windows NT4.0.

Under Windows 9x and Windows 2000

For these Operating Systems, a specific library called IP Helper library is available and allows the capability to retrieve easily all preceding parameters.

Under Windows NT

For Windows NT4.0, the way to retrieve IP configuration is different than for Windows 9x and Windows 2000. Some of the functions and structures from IP Helper library are not supported under Windows NT4.0 and we must search for all the values inside the registry.

Here are the main steps for correctly detecting IP configuration on a local system:

1. Retrieve number of Network Adapters
2. For each Network Adapter, do
   a. Retrieve the "ServiceName" key value attached to the current Network adapter
   b. If (EnableDHCP == 1) then Retrieve the following values:
      i. DhcpIPAddress
      ii. DhcpSubnetMask
      iii. DhcpDefaultGateway

      } Registry keys
   c. Else
      i. IPaddress
      ii. SubnetMask
      iii. DefaultGateway

      } Registry keys

Below are all registry paths where all keys are stored:

1. "ServiceName" key value
   a. HKLM\SOFTWARE\Microsoft\Windows\NT\CurrentVersion\NetworkCards\1
2. EnableDHCP and all other IP key values
   a. HKLM\SYSTEM\CurrentControlSet\Services\{ServiceName}\\Parameters\tcpip

Modem test

Implementation aspects

This diagnostic shall have two phases. The first phase shall be responsible for locating the corresponding driver and "COM" port for the modem device. The second part shall be responsible for sending a couple of Hayes compatible AT commands to the modem device.

- First phase: This phase shall search the registry for a corresponding modem device and the associate port. The search, which shall be done in Windows registry, shall take into account the current Operating System, since the location of the modem device varies according to it. The absence of the referred information in Windows registry shall indicate an error condition, since the modem is present, but not properly configured.

- Second phase: This phase shall be responsible for issuing Hayes compatible commands to the modem device. An incorrect answer from the modem may indicate an error in the device or in the corresponding device driver configuration. It shall also attempt to detect the dial tone. If it cannot be found, a warning message shall be displayed (instead of an error message).

Action flow

Variables, constants and structures

Name	Description
SzModemResponse	String that stores the value from the modem read thread.
szModemInput	String that stores data read continuously from the modem.
BaudRate	Defines the current baud rate setting

Table 11. Modem variables, constants and structures

Determining the modem "COM1" port (1st phase)

1. Get OS
2. If OS = Windows 2000 or XP (all sub-version)
   a. Search registry key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Enum\PCI\<X>\<Y> where value class="Modem"
      i. If the key cannot be found, set error ERROR_MODEM_DRIVER
   b. Under the located key "HKEY_LOCAL_MACHINE\System\CurrentControlSet\Enum\PCI\<X>\<Y>\ Device Parameters" store the contents of value "PortName" to szPortName.
      i. If the value cannot be found, set error ERROR_MODEM_PORT1
3. Else If OS = Windows 95, 98, Me (all sub-versions)
   a. Search registry key HKEY_LOCAL_MACHINE\Enum\PCI\<X>\<Y> where value class="Modem"
      i. If the key cannot be found, set error ERROR_MODEM_DRIVER2
   b. Under the located key key "HKEY_LOCAL_MACHINE\ Enum\PCI\<X>\<Y>" store the contents of value "PORTNAME" to szPortName.

i. If the value cannot be found, set error ERROR_MODEM_PORT2

4. Else set error ERROR_OS_UNSUPPORTED

Sending AT commands to the modem (2nd phase)

1. Use CreateFile() to open the serial port szPortName concatenated with ":". Use option OPEN_EXISTING for dwCreationDisposition and 0 for dwFlagsAndAttributes.
   a. In case of error set error to ERROR_DEVICE_OPEN
2. Create thread T1 (Specified below)
3. Use API function GetCommState() to get the serial port state
   a. In case of error set error to ERROR_DEVICE_GETSTATE
4. Varying baud rates [BaudRate=9600, 14400, 19200, 28800, 57600] do
   a. Set the serial port baud rate to BaudRate using SetCommState and the configuration obtained with GetCommState()
      i. In case of error set error to ERROR_DEVICE_SETCOMM<BaudRate>
   b. Write to port command "AT"+CR
      i. If local variable szModemResponse is empty, set error to ERROR_MODEM_RESPONSE_<BaudRate>
      ii. Else If local variable szModemResponse does not contain string "OK", set error to ERROR_MODEM_RESPONSE2_<BaudRate>
   c. Write to port command "ATZ"+CR
      i. If local variable szModemResponse is empty set error to ERROR_MODEM_RESPONSE3_<BaudRate>
      ii. Else If local variable szModemResponse does not contain string "OK", set error to ERROR_MODEM_Reset_<BaudRate>
   d. Write to port command "ATI1"+CR
      i. If local variable szModemResponse is empty set error to ERROR_MODEM_RESPONSE4_<BaudRate>
      ii. Else If local variable szModemResponse does not contain string "OK", set error to ERROR_MODEM_CHECKSUM_<BaudRate>
   e. Write to port command "AT+FCLASS"
      i. In case of error set error to RROR_MODEM_RESPONSE5_<BaudRate>
      ii. Else If local variable szModemResponse does not contain string "OK" set warning message WARNING_MODEM_NOFAX
5. Resume and Destroy thread T1
6. Close port handles

Detailed specification for WritePort(szCommand)

1. Enter Critical Section C1
2. Clean szModemInput
3. Leave Critical Section C1
4. Send command "AT" + CR to the port using WriteFile( )

5. Wait for TIMEOUT_WRITE seconds using Sleep API function
6. Enter Critical Section C1
7. Copy szModemInput to szModemResponse
8. Leave Critical Section C1

Detailed specification for thread T1

This thread should be implemented as the readPort() function from the serial port prototype application. Follows the steps executed by the thread:

1. Do while the port handle is valid
2. Wait for a serial port event (WaitCommEvent( ))
3. If event=character received then
   a. Set dwTransferedCharacters to 1
4. While dwTransferedCharacters>0
   a. Read 1 character from port handle and store it to temporary buffer Copy szModemInput to szModemResponse
5. Begin Critical Section C1
6. Append the character read into temporary buffer to szModemInput
7. Leave Critical Section C1

Error codes and returns

Error code numbers shall be defined later in the system context. Errors detailed here are defined in constants, although they have been not yet mapped to error codes numbers.

Errors
ERROR_MODEM_PORT1
ERROR_MODEM_DRIVER2
ERROR_MODEM_PORT2
ERROR_OS_UNSUPPORTED
ERROR_DEVICE_OPEN
ERROR_DEVICE_GETSTATE
ERROR_DEVICE_SETCOM <BaudRate>
ERROR_MODEM_RESPONSE_<BaudRate>
ERROR_MODEM_RESPONSE2_<BaudRate>
ERROR_MODEM_RESPONSE3_<BaudRate>
ERROR_MODEM_Reset_<BaudRate>
ERROR_MODEM_RESPONSE4_<BaudRate>
ERROR_MODEM_CHECKSUM_<BaudRate>
ERROR_MODEM_RESPONSE5_<BaudRate>
WARNING_MODEM_NOFAX

Table 12. Modem errors

Mouse Test

Main failures

1. Interrupt conflict
2. Address conflict
3. Driver version not correct or absent
4. Connectivity problem
5. Damaged mouse

Supported models

1. 2 buttons
2. 3 buttons
3. 2 buttons + wheel
4. 2 buttons + cordless wheel

Failure detection

1. Check driver version
2. Check mouse configuration
3. Verify connectivity
4. Check mouse behavior

Note: mouse diagnostic does not configure the mouse.

Test algorithm

if (the mouse is not present)

"verify if your mouse is connected"

"if it is ok change your mouse"

else

identification of the mouse:

"display mice pictures to help the User to chose his mouse type"

Mouse test

if the choosen mouse is bad configured (registry)

"your mouse is not well configured"

we propose a shortcut for control panel

if the version of the driver is not correct

"you should update the driver of the mouse, you could have some problems with certain software"

Position test

1. Mouse configuration is displayed (wheel activated, right or left click)
2. User checks the click of buttons
3. User checks the behavior of the wheel
4. User can verify that the position of the mouse changes

Double-click test

1. The speed of the double click is shown.
2. If the test fails, the speed is modified (medium) and the user tests again.
3. If the test succeeds, we propose a shortcut for the control panel.

USB test

USB cannot be tested without a device. There are two components, the USB Host Controller, and the Root HUB. Their main functions are not used because there is no activity on the USB.

Components tested	Description of test
CONTROLLER	Get the Driver Name used by the OS
ROOT HUB (Ports)	Get the Port Status Get the main characteristics of the ROOT HUB

Table 13. USB components

With one or several devices, it is possible to communicate with devices and to get information. If most of the descriptors are returned correctly, then the USB bus is functional.

Algorithms

Objective

The main purpose is to communicate with the device, and to demonstrate that USB is functional. The communication procedure is:

1. Open a pipe with the USB Hub Driver,
2. Detect a device or several devices,
3. Send several queries, and receive the result.

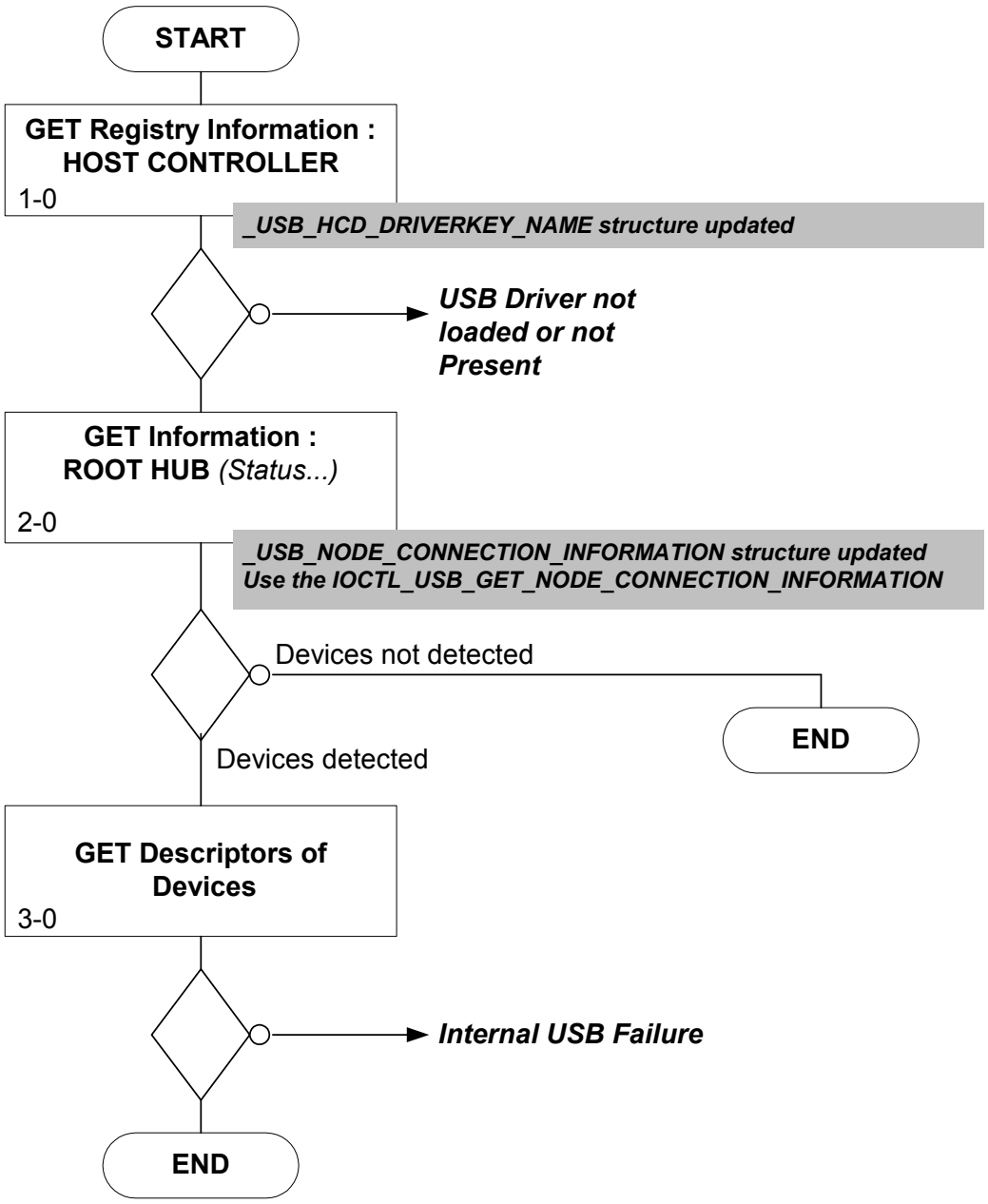

Figure 4. USB general algorithm

Get registry information

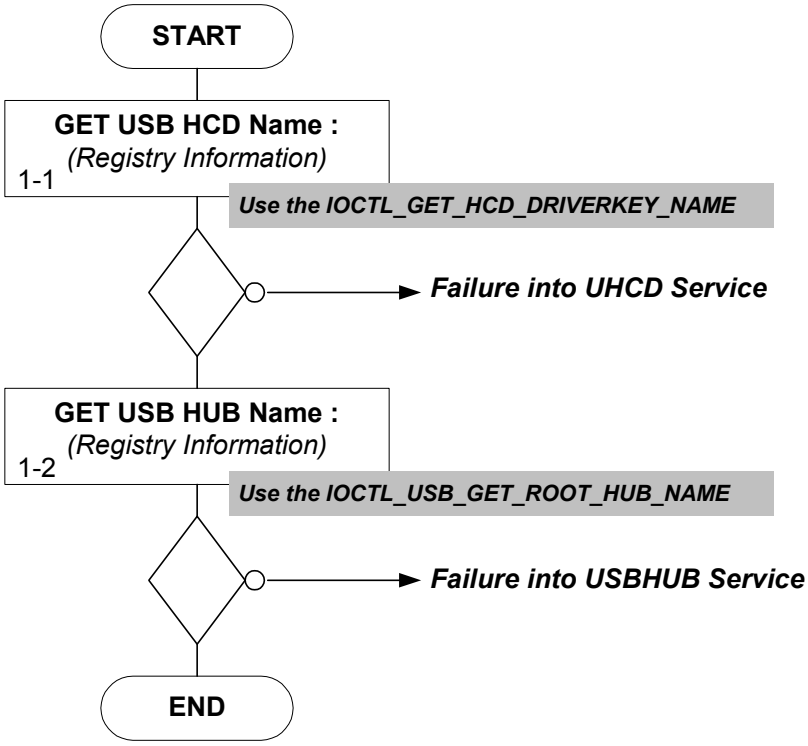

Figure 5. USB get registry information

Input

(Use the usbioctl header file)

- IOCTL_GET_HCD_DRIVERKEY_NAME : This code allows to get the Driver Name into the NT registry.
- IOCTL_GET_ROOT_HUB_NAME : This code allows to get the Root Hub Name.

Output

(Use the usbioctl header file)

- USB_HCD_DRIVERKEY_NAME structure : This structure contains :
  - The Host Controller Driver,
  - The number of characters of the Driver Name.

Errors and warnings

Error code	Description	Resolution
ERR_USB_UHCD_SERVICE_FAILURE	Failure into UHCD Service	Reinstall the UHCD service
ERR_USB_USBHUB_SERVICE_FAILURE	Failure into USBHUB Service	Reinstall the USBHUB Service

Table 14. USB errors and warnings

Get root hub information

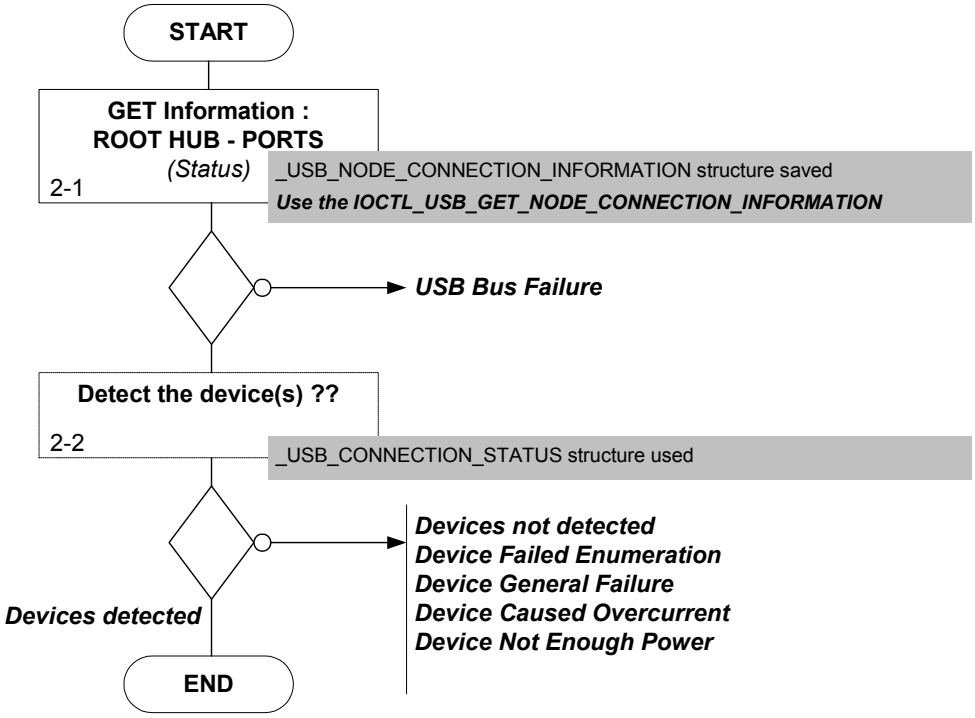

Figure 6. USB get root hub information

Input

(Use the usbioctl header file)

- OCTL_USB_GET_NODE_CONNECTION_INFORMATION : This code is used to launch a GET_STATUS command to the Internal ROOT HUB.

Output

(Use the usbioctl header file)

- _USB_HUB_DESCRIPTOR : This structure contains :
  - Length of this descriptor
  - Hub configuration type
  - Number of ports on this hub
  - Hub Characteristics
  - Port power on until power good in 2ms
  - Max current in mA
- _USB_CONNECTION_STATUS : This structure contains :
  - No Device is connected,
  - Device connected on a port,
  - The OS Enumeration process failed,
  - General Failure of Device is detected by the OS,

- An Over-current, caused by the Device, is detected,
- The Device has not enough power,
- The Device has not enough bandwidth.

Errors and warnings

Error code	Description	Resolution
ERR_USB_BUS_FAILURE	USB Bus Failure	• Reinstall the USBHUB service • Unplug and plug a device
ERR_USB_DEVICE_NOT_CONNECTED	Device not Detected	**Warning**: Plug a device to check, correctly into the USB Bus.
ERR_USB_OS_FAILED_ENUM	Device Failed Enumeration (Generate by the Operating System)	• Unplug and plug the device • Or reinstall the USBHUB service
ERR_USB_OS_GENERAL_FAILURE	Device General Failure (Generate by the Operating System)	• Unplug and plug the device • Breakdown of the Device
ERR_USB_OS_CAUSED_OVERCUR	Device Caused Overcurrent (Generate by the Operating System)	• Unplug and plug the device • Failure with the Device
ERR_USB_OS_NOT_ENOUGH_POWER	Device does not contain enough Power (generated by the Operating System)	• Unplug and plug the device with an other USB port • Failure into the Root Hub or the Device

Table 15. USB get root hub errors and warnings

Get device descriptors

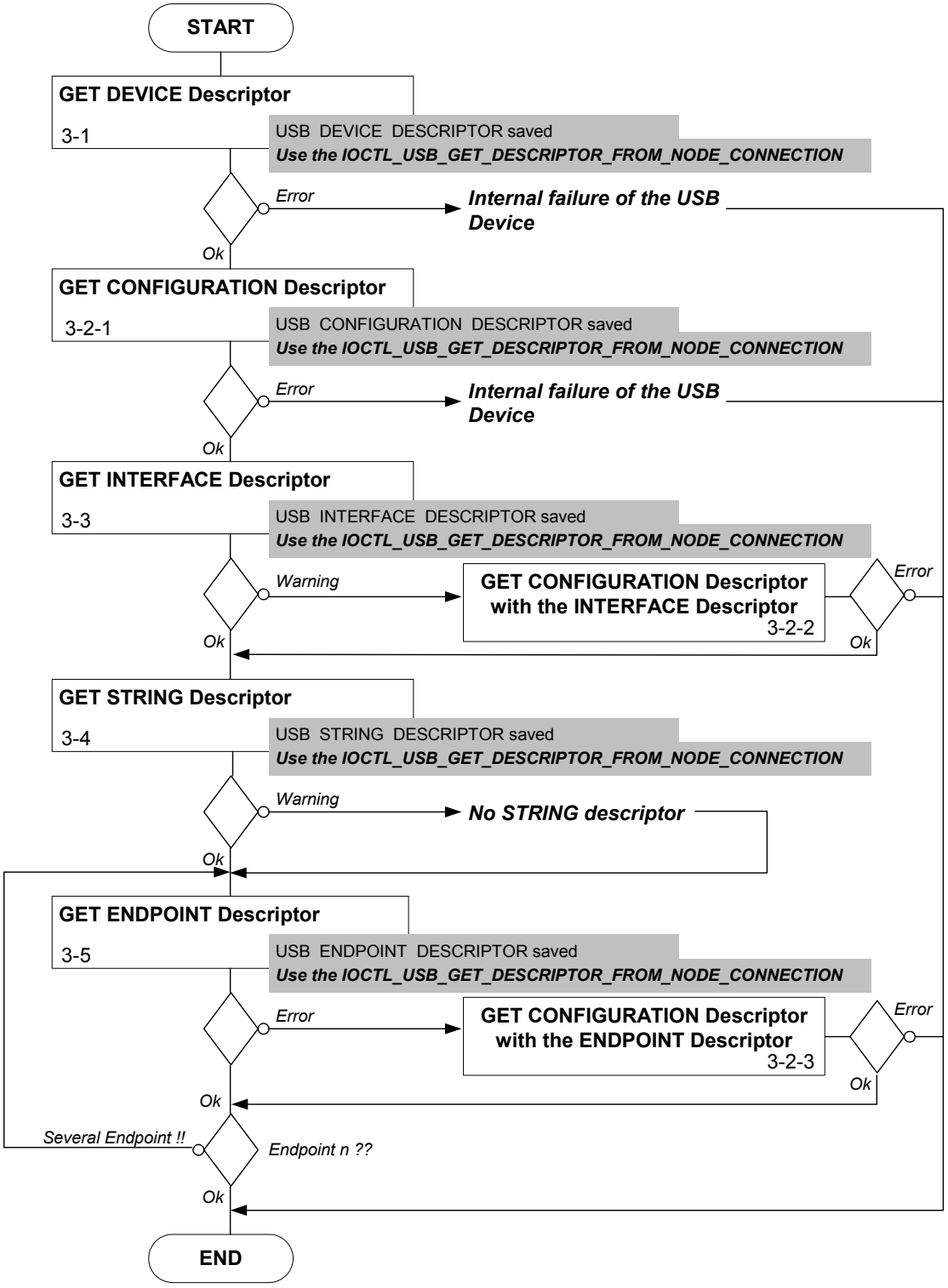

Figure 7. USB get device descriptors

Input

(Use the usbioctl header file)

- _USB_DESCRIPTOR_REQUEST : This structure contains :
  - Characteristics of request
  - Specific Request
  - Parameter of request
  - Number of Data
  - Data
- IOCTL_USB_GET_DESCRIPTOR_FROM_NODE_CONNECTION : This code is used to launch a GET_DESCRIPTOR command to device, with a specific type.

Output

(Use the usbioctl header file) And (Use the usb100 header file)

- _USB_DEVICE_DESCRIPTOR
- _USB_CONFIGURATION_DESCRIPTOR
- _USB_INTERFACE_DESCRIPTOR
- _USB_STRING_DESCRIPTOR
- _USB_ENDPOINT_DESCRIPTOR

These structures are defined in the Standard USB Descriptor Specifications.

Errors and warnings

Error code	Description	Resolution
ERR_USB_DEVICE_DRIVER_FAILURE	USB driver for a USB device is not correctly installed	Install or reinstall the USB device driver
ERR_USB_DEVICE_DESC_FAILURE		
ERR_USB_CONFIGURATION_DESC_FAILURE		
ERR_USB_INTERFACE_DESC_FAILURE		
ERR_USB_ENDPOINT_FAILURE	Internal failure of the USB Device	▪ Unplug and plug the device ▪ Verify the USB cable/ USB connector ▪ Select an other USB connector
ERR_NOT_ENOUGH_MEMORY		Free memory
ERR_USB_NO_CONTROLLER_FOUND	No USB controller found	
ERR_USB_CONTROLLER_NOT_IDENTIFY	The USB controller is not identified	

Table 16. USB get device descriptors errors and warnings

For more information

To learn more about HP Instant Support Corporate Edition, visit **www.hp.com/go/instantsupport**.

5982-3708EN, 09/2003

invent