— APPENDIX A —

FUNCTIONS BY OPCODE

GEMDOS Functions by Opcode

Dec	Hex	Function	Summary	Page
0	0x00	**PtermØ()**	Exit process with a return code of 0.	2.122
1	0x01	**Cconin()**	Fetch a character from the console device and echo it.	2.41
2	0x02	**Cconout()**	Output a character to the console device processing any special keys.	2.43
3	0x03	**Cauxin()**	Fetch character from the auxiliary device.	2.39
4	0x04	**Cauxout()**	Output a character to the auxiliary device.	2.41
5	0x05	**Cprnout()**	Output a character to the printer device.	2.47
6	0x06	**Crawio()**	Perform input and output on the console device.	2.49
7	0x07	**Crawcin()**	Output a character to the console device.	2.48
8	0x08	**Cnecin()**	Fetch a character from the console device.	2.46
9	0x09	**Cconws()**	Write a string to the console device.	2.45
10	0x0A	**Cconrs()**	Read a string from the console device.	2.44
11	0x0B	**Cconis()**	Determine if a character is waiting to be received from the console device.	2.42
14	0x0E	**Dsetdrv()**	Set the default drive.	2.62
16	0x10	**Cconos()**	Determine if a character may be sent to the console device.	2.43
17	0x11	**Cprnos()**	Determine if a character may be sent to the printer device.	2.46
18	0x12	**Cauxis()**	Determine if a character is waiting to be received from the auxiliary device.	2.39
19	0x13	**Cauxos()**	Determine if a character may be sent to the auxiliary device.	2.40
20	0x14	**Maddalt()**	Notify **GEMDOS** of additional memory.	2.97
25	0x19	**Dgetdrv()**	Return the default drive.	2.56
26	0x1A	**Fsetdta()**	Set the address of the **DTA**.	2.91
32	0x20	**Super()**	Modify user/supervisor status.	2.128
42	0x2A	**Tgetdate()**	Get the current date.	2.132
43	0x2B	**Tsetdate()**	Set the current date.	2.133
44	0x2C	**Tgettime()**	Get the current time.	2.132
45	0x2D	**Tsettime()**	Set the current time.	2.133
47	0x2F	**Fgetdta()**	Return a pointer to the **DTA**.	2.79
48	0x30	**Sversion()**	Obtain the current **GEMDOS** version.	2.129
49	0x31	**Ptermres()**	Exit process leaving some data intact.	2.123
54	0x36	**Dfree()**	Determine the free space on a drive.	2.54
57	0x39	**Dcreate()**	Create a directory.	2.53
58	0x3A	**Ddelete()**	Delete a directory.	2.54
59	0x3B	**Dsetpath()**	Set the default path.	2.63
60	0x3C	**Fcreate()**	Create a file.	2.74
61	0x3D	**Fopen()**	Open a file.	2.84
62	0x3E	**Fclose()**	Close a file.	2.66
63	0x3F	**Fread()**	Read binary data from a file.	2.87
64	0x40	**Fwrite()**	Write binary data to a file.	2.95
65	0x41	**Fdelete()**	Delete a file.	2.76
66	0x42	**Fseek()**	Move a file pointer.	2.89
67	0x43	**Fattrib()**	Get or set the attributes of a file.	2.64
68	0x44	**Mxalloc()**	Allocate memory with preference.	2.100
69	0x45	**Fdup()**	Duplicate a file handle.	2.76

A.4 – Functions by Opcode

Dec	Hex	Function	Summary	Page
70	0x46	**Fforce()**	Redirect one handle to another.	2.77
71	0x47	**Dgetpath()**	Return the default path.	2.57
72	0x48	**Malloc()**	Allocate memory.	2.98
73	0x49	**Mfree()**	Free allocated memory.	2.99
74	0x4A	**Mshrink()**	Shrink or expand a block of memory.	2.99
75	0x4B	**Pexec()**	Execute another process.	2.103
76	0x4C	**Pterm()**	Exit process with the specified return code.	2.121
78	0x4E	**Fsfirst()**	Find a file with the specified mask.	2.92
79	0x4F	**Fsnext()**	Find subsequent files with the specified mask.	2.93
86	0x56	**Frename()**	Rename a file or directory.	2.89
87	0x57	**Fdatime()**	Get or set the time/date flags of a file.	2.75
92	0x5C	**Flock()**	Set or remove a file lock.	2.82
255	0xFF	**Syield()**	Surrender the remaining portion of the processes timeslice.	2.130
256	0x100	**Fpipe()**	Establish a communication pipeline between processes.	2.86
260	0x104	**Fcntl()**	Perform a file-system specific file operation.	2.67
261	0x105	**Finstat()**	Determine the input status of a file.	2.80
262	0x106	**Foutstat()**	Determine the output status of a file.	2.85
263	0x107	**Fgetchar()**	Get a character from a file.	2.79
264	0x108	**Fputchar()**	Output a character to a file.	2.86
265	0x109	**Pwait()**	Determine the exit code of a stopped or terminated child process.	2.125
266	0x10A	**Pnice()**	Alter the process priority of the calling process.	2.111
267	0x10B	**Pgetpid()**	Obtain the process ID of the calling process.	2.107
268	0x10C	**Pgetppid()**	Obtain the process ID of the processes' parent.	2.108
269	0x10D	**Pgetpgrp()**	Obtain the process group ID of the calling process.	2.107
270	0x10E	**Psetpgrp()**	Set the process group ID for the calling process.	2.115
271	0x10F	**Pgetuid()**	Obtain the user ID of the calling process.	2.108
272	0x110	**Psetuid()**	Set the user ID for the calling process.	2.116
273	0x111	**Pkill()**	Send a signal to one or more processes.	2.109
274	0x112	**Psignal()**	Determine the action to take when a signal is received.	2.118
275	0x113	**Pvfork()**	Create a duplicate of the current process which shares address and data space with its parent.	2.124
276	0x114	**Pgetgid()**	Obtain the group ID of the calling process.	2.107
277	0x115	**Psetgid()**	Set the group ID of the calling process.	2.114
278	0x116	**Psigblock()**	Block selected signals from delivery.	2.118
279	0x117	**Psigsetmask()**	Specifies which signals should be blocked and which should be received.	2.121
280	0x118	**Pusrval()**	Get or set the user-defined value associated with a process.	2.124
281	0x119	**Pdomain()**	Get or set the processes execution domain.	2.102
282	0x11A	**Psigreturn()**	Clean up from a signal handler.	2.120
283	0x11B	**Pfork()**	Create a copy of the current process.	2.105
284	0x11C	**Pwait3()**	Determine the exit code of stopped or terminated child processes.	2.126
285	0x11D	**Fselect()**	Enumerate file descriptors which are ready for reading/writing.	2.90
286	0x11E	**Prusage()**	Return resource usage information on the calling process.	2.112
287	0x11F	**Psetlimit()**	Read or modify resource usage limits for a process.	2.114
288	0x120	**Talarm()**	Read or set an alarm for the current process.	2.131

Dec	Hex	Function	Summary	Page
289	0x121	**Pause()**	Suspend the process until a signal is received.	2.101
290	0x122	**Sysconf()**	Return information regarding current capabilities and limitations of processes running under **MiNT**.	2.130
291	0x123	**Psigpending()**	Determines which signals have been sent but not yet received to the calling process.	2.120
292	0x124	**Dpathconf()**	Return information regarding limitations and capabilities of a file system.	2.59
293	0x125	**Pmsg()**	Send or receive a message.	2.109
294	0x126	**Fmidipipe()**	Change the file handles which refer to MIDI input and output.	2.83
295	0x127	**Prenice()**	Alter the process priority of the specified process.	2.111
296	0x128	**Dopendir()**	Open a directory.	2.58
297	0x129	**Dreaddir()**	Read a directory entry.	2.61
298	0x12A	**Drewinddir()**	Reset the directory pointer.	2.62
299	0x12B	**Dclosedir()**	Close a directory.	2.50
300	0x12C	**Fxattr()**	Return extended attribute information for a file.	2.95
301	0x12D	**Flink()**	Create a file link.	2.81
302	0x12E	**Fsymlink()**	Establish a symbolic link to a file.	2.94
303	0x12F	**Freadlink()**	Determine the actual file to which a link refers.	2.88
304	0x130	**Dcntl()**	Perform a file-system specific device operation.	2.50
305	0x131	**Fchown()**	Modify the ownership of a file.	2.66
306	0x132	**Fchmod()**	Modify the access permission flags of a file.	2.65
307	0x133	**Pumask()**	Determines the minimum file and/or directory creation access permission masks.	2.123
308	0x134	**Psemaphore()**	Create a semaphore.	2.113
309	0x135	**Dlock()**	Lock or unlock a **BIOS** disk device.	2.57
310	0x136	**Psigpause()**	Suspends the process until a specified signal (or signals) is received.	2.119
311	0x137	**Psigaction()**	Changes the way a signal is handled.	2.116
312	0x138	**Pgeteuid()**	Returns the effective user ID of the caller.	2.106
313	0x139	**Pgetegid()**	Returns the effective group ID of the caller.	2.106
314	0x13A	**Pwaitpid()**	Attempts to determine the exit code of a particular process.	2.127
315	0x13B	**Dgetcwd()**	Returns the current **GEMDOS** working directory for the process on the specified drive.	2.56
316	0x13C	**Salert()**	Sends an alert to the alert pipe ‘U:\PIPE\ALERT’.	2.128

BIOS Functions by Opcode

Dec	Hex	Function	Summary	Page
0	0x00	**Getmpb()**	Return the address of the **MPB** (Memory Parameter Block) structure.	3.31
1	0x01	**Bconstat()**	Determine if a character is waiting from a device.	3.28
2	0x02	**Bconin()**	Input a character from a device.	3.27
3	0x03	**Bconout()**	Output a character from a device.	3.28
4	0x04	**Rwabs()**	Read/write sectors to a device.	3.34
5	0x05	**Setexc()**	Set or read a system exception vector.	3.35
6	0x06	**Tickcal()**	Return the current system timer calibration.	3.36
7	0x07	**Getbpb()**	Return the address of the **BPB** (**BIOS** Parameter Block).	3.30
8	0x08	**Bcostat()**	Determine if a device is ready to receive a character.	3.29
9	0x09	**Mediach()**	Determine if a drive’s media has been changed.	3.33
10	0x0A	**Drvmap()**	Return a bitmap of mounted drives.	3.30
11	0x0B	**Kbshift()**	Return the state of the keyboard shift keys.	3.32

XBIOS Functions by Opcode

Dec	Hex	Function	Summary	Page
0	0x00	**Initmous()**	Initialize the mouse handler.	4.73
1	0x01	**Ssbrk()**	Reserve memory at the top of RAM.	4.102
2	0x02	**Physbase()**	Return the address of the physical screen.	4.85
3	0x03	**Logbase()**	Return the address of the logical screen.	4.80
4	0x04	**Getrez()**	Return the current screen resolution code.	4.68
5	0x05	**Setscreen()** and **VsetScreen()**	Set the current screen address and mode.	4.97 4.108
6	0x06	**Setpalette()**	Set entries in the ST compatible palette.	4.95
7	0x07	**Setcolor()**	Set an entry in the ST compatible palette.	4.93
8	0x08	**Floprd()**	Read a sector from a floppy disk.	4.66
9	0x09	**Flopwr()**	Write a sector to a floppy disk.	4.67
10	0x0A	**Flopfmt()**	Format a sector on a floppy disk.	4.63
11	0x0B	**Dbmsg()**	Send a debugging message to the resident debugger.	4.28
12	0x0C	**Midiws()**	Write a string to the MIDI port.	4.82
13	0x0D	**Mfpint()**	Define an MFP interrupt.	4.81
14	0x0E	**Iorec()**	Return the address of the system **IOREC** structure.	4.75
15	0x0F	**Rsconf()**	Configure the currently mapped RS-232 port.	4.89
16	0x10	**Keytbl()**	Return the addresses of the current key mapping tables.	4.78
17	0x11	**Random()**	Return a random number.	4.89
18	0x12	**Protobt()**	Prototype a floppy boot sector.	4.86
19	0x13	**Flopver()**	Verify a sector on a floppy disk.	4.66
20	0x14	**Scrdump()**	Execute the built-in screen dump code.	4.91
21	0x15	**Cursconf()**	Configure the **TOS** cursor.	4.27
22	0x16	**Settime()**	Set the time of day and current date.	4.98
23	0x17	**Gettime()**	Get the time of day and current date.	4.69
24	0x18	**Bioskeys()**	Reset the keyboard mapping tables to default.	4.24
25	0x19	**Ikbdws()**	Write a string to the intelligent keyboard controller.	4.72
26	0x1A	**Jdisint()**	Disable an MFP interrupt.	4.76
27	0x1B	**Jenabint()**	Enable an MFP interrupt.	4.76
28	0x1C	**Giaccess()**	Modify or set a register on the PSG.	4.70
29	0x1D	**Offgibit()**	Toggle bits of the PSG Port A off.	4.84
30	0x1E	**Ongibit()**	Toggle bits of the PSG Port A on.	4.84
31	0x1F	**Xbtimer()**	Set an interrupt on the 68901.	4.113
32	0x20	**Dosound()**	Start an interrupt driven sound routine.	4.33
33	0x21	**Setprt()**	Set or read the printer configuration bits.	4.96
34	0x22	**Kbdvbase()**	Return the address of the current IKBD interrupt table.	4.77
35	0x23	**Kbrate()**	Set or read the keyboard repeat rate.	4.78
36	0x24	**Prtblk()**	Print a block of memory using the built-in screen dump routines.	4.87
37	0x25	**Vsync()**	Hold the process until the next vertical blank.	4.110
38	0x26	**Supexec()**	Execute a routine in supervisor mode.	4.103
39	0x27	**Puntaes()**	Discard the **AES**.	4.88

A.10 – Functions by Opcode

Dec	Hex	Function	Summary	Page
41	0x29	**Floprate()**	Set the floppy drive seek rates.	4.65
42	0x2A	**DMAread()**	Read sectors from a DMA/SCSI device.	4.31
43	0x2B	**DMAwrite()**	Write sectors to a DMA/SCSI device.	4.32
44	0x2C	**Bconmap()**	Modify the **BIOS** device mapping table.	4.23
46	0x2E	**NVMaccess()**	Access non-volatile RAM.	4.83
48	0x30	**Metainit()**	Initialize **MetaDOS**.	4.80
64	0x40	**Blitmode()**	Get or set the state of the BLiTTER chip.	4.25
80	0x50	**EsetShift()**	Set the TT030 shift mode registers.	4.61
81	0x51	**EgetShift()**	Get the TT030 shift mode registers.	4.57
82	0x52	**EsetBank()**	Set the current TT030 color bank.	4.58
83	0x53	**EsetColor()**	Get or set a color in the TT030 palette.	4.59
84	0x54	**EsetPalette()**	Set the TT030 palette.	4.60
85	0x55	**EgetPalette()**	Get the TT030 palette.	4.56
86	0x56	**EsetGray()**	Set the TT030 gray mode register.	4.60
87	0x57	**EsetSmear()**	Set the TT030 smear mode register.	4.62
88	0x58	**VsetMode()**	Set the Falcon030 video mode.	4.107
89	0x59	**VgetMonitor()**	Identify the kind of monitor attached to the Falcon030.	4.104
90	0x5A	**VsetSync()**	Set the Falcon030 sync mode.	4.109
91	0x5B	**VgetSize()**	Get the size of screen memory in bytes.	4.105
92	0x5C	**VsetMask()**	Set the mask assigned to each true color plotted.	4.106
93	0x5D	**VsetRGB()**	Set the Falcon030 palette using RGB data.	4.108
94	0x5E	**VgetRGB()**	Get the Falcon030 palette using RGB data.	4.104
96	0x60	**Dsp_DoBlock()**	Transfer bytewise packed data to/from the DSP.	4.38
97	0x61	**Dsp_BlkHandshake()**	Handshakes bytewise packed data to/from the DSP.	4.35
98	0x62	**Dsp_BlkUnpacked()**	Transfers data stored in a longword array to/from the DSP.	4.36
99	0x63	**Dsp_InStream()**	Transfers data to the DSP via an interrupt handler.	4.45
100	0x64	**Dsp_OutStream()**	Transfers data from the DSP via an interrupt handler.	4.51
101	0x65	**Dsp_IOStream()**	Transfers data to/from the DSP via concurrent interrupt handlers.	4.46
102	0x66	**Dsp_RemoveInterrupts()**	Disable the generation of DSP interrupts.	4.51
103	0x67	**Dsp_GetWordSize()**	Get the current size of a DSP word.	4.41
104	0x68	**Dsp_Lock()**	Lock the DSP system.	4.48
105	0x69	**Dsp_Unlock()**	Unlock the DSP system.	4.55
106	0x6A	**Dsp_Available()**	Determines the amount of free X and Y memory available in the DSP.	4.34
107	0x6B	**Dsp_Reserve()**	Reserves a portion of DSP memory for a user program	4.53
108	0x6C	**Dsp_LoadProg()**	Loads a '.LOD' file from disk, transmits it to the DSP, and executes it.	4.47
109	0x6D	**Dsp_ExecProg()**	Transfers a DSP program in memory to the DSP and executes it.	4.39
110	0x6E	**Dsp_ExecBoot()**	Resets the DSP and loads a new bootstrap program into the first 512 words of DSP memory.	4.39

Dec	Hex	Function	Summary	Page
111	0x6F	**Dsp_LodToBinary()**	Converts a '.LOD' file to binary format.	4.49
112	0x70	**Dsp_TriggerHC()**	Causes a host command set aside for DSP programs to execute.	4.55
113	0x71	**Dsp_RequestUniqueAbility()**	Requests a unique DSP ability identifier.	4.52
114	0x72	**Dsp_GetProgAbility()**	Returns the ability code for the program residing in DSP memory.	4.40
115	0x73	**Dsp_FlushSubroutines()**	Removes all DSP subroutines from memory.	4.40
116	0x74	**Dsp_LoadSubroutine()**	Loads a DSP subroutine into memory.	4.48
117	0x75	**Dsp_InqSubrAbility()**	Determines if a subroutine with the specified ability code is currently loaded into the DSP.	4.44
118	0x76	**Dsp_RunSubroutine()**	Begins execution of the specified subroutine.	4.53
119	0x77	**Dsp_Hf0()**	Reads/writes bit #3 of the HSR.	4.41
120	0x78	**Dsp_Hf1()**	Reads/writes bit #4 of the HSR.	4.42
121	0x79	**Dsp_Hf2()**	Reads bit #5 of the HSR.	4.43
122	0x7A	**Dsp_Hf3()**	Reads bit #6 of the HSR.	4.43
123	0x7B	**Dsp_BlkWords()**	Transfers an array of **WORD**s to/from the DSP.	4.37
124	0x7C	**Dsp_BlkBytes()**	Transfers an array of bytes to/from the DSP.	4.34
125	0x7D	**Dsp_Hstat()**	Returns the value of the DSP's ICR register.	4.44
126	0x7E	**Dsp_SetVectors()**	Defines interrupt handlers to be called when DSP data is ready to be sent or received.	4.54
127	0x7F	**Dsp_MultBlocks()**	Transmits multiple blocks to/from the DSP.	4.50
128	0x80	**Locksnd()**	Lock the sound system.	4.79
129	0x81	**Unlocksnd()**	Unlock the sound system.	4.103
130	0x82	**Soundcmd()**	Execute a sound system specific function.	4.100
131	0x83	**Setbuffer()**	Set the record and playback buffers.	4.92
132	0x84	**Setmode()**	Set the playback/record mode.	4.94
133	0x85	**Settracks()**	Set the playback/record tracks.	4.99
134	0x86	**Setmontracks()**	Set the track to be output over the speaker/headphone.	4.95
135	0x87	**Setinterrupt()**	Set the sound system interrupts.	4.93
136	0x88	**Buffoper()**	Enable or disable playback/recording.	4.25
137	0x89	**Dsptristate()**	Connect or disconnect the DSP from the connection matrix.	4.56
138	0x8A	**Gpio()**	Read or write data over the general purpose pins on the DSP port.	4.72
139	0x8B	**Devconnect()**	Connect devices in the connection matrix.	4.29
140	0x8C	**Sndstatus()**	Obtain the status of the sound system.	4.99
141	0x8D	**Buffptr()**	Return the current position of the record or playback buffer pointers.	4.26
165	0xA5	**WavePlay()**	Playback a DMA sample.	4.110

AES Functions by Opcode

Dec	Hex	Function	Summary	Page
10	0x0A	**appl_init()**	Initializes a **GEM** application.	6.53
11	0x0B	**appl_read()**	Reads data from the message pipe.	6.54
12	0x0C	**appl_write()**	Writes data to the message pipe.	6.58
13	0x0D	**appl_find()**	Locates a system process.	6.47
14	0x0E	**appl_tplay()**	Plays back recorded events.	6.56
15	0x0F	**appl_trecord()**	Records keyboard and mouse events.	6.57
18	0x12	**appl_search()**	Enumerates system processes.	6.55
19	0x13	**appl_exit()**	Prepares a **GEM** application for termination.	6.47
20	0x14	**evnt_keybd()**	Waits for a keyboard event.	6.63
21	0x15	**evnt_button()**	Waits for a mouse button event.	6.61
22	0x16	**evnt_mouse()**	Waits for a mouse rectangle event.	6.70
23	0x17	**evnt_mesag()**	Waits for an application message.	6.64
24	0x18	**evnt_timer()**	Waits for a timer event.	6.73
25	0x19	**evnt_multi()**	Waits for multiple events.	6.71
26	0x1A	**evnt_dclick()**	Sets the mouse double-click rate.	6.62
30	0x1E	**menu_bar()**	Displays/removes a menu bar.	6.105
31	0x1F	**menu_icheck()**	Places a checkmark beside a menu item.	6.106
32	0x20	**menu_ienable()**	Enables/disables a menu item.	6.106
33	0x21	**menu_tnormal()**	Selects/deselects a menu item or title.	6.111
34	0x22	**menu_text()**	Changes menu item/title text.	6.111
35	0x23	**menu_register()**	Registers applications in the menu bar.	6.109
36	0x24	**menu_popup()**	Manages a floating popup menu.	6.108
37	0x25	**menu_attach()**	Attaches a sub-menu to a menu item.	6.103
38	0x26	**menu_istart()**	Defines the initial selection of a sub-menu.	6.107
39	0x27	**menu_settings()**	Modifies popup menu settings.	6.110
40	0x28	**objc_add()**	Adds an object to an object tree.	6.115
41	0x29	**objc_delete()**	Deletes an object from an object tree.	6.116
42	0x2A	**objc_draw()**	Draws an object tree.	6.117
43	0x2B	**objc_find()**	Locates an object based on screen coordinates.	6.119
44	0x2C	**objc_offset()**	Determines the offset of child objects in an object tree.	6.120
45	0x2D	**objc_order()**	Reorders objects within an object tree.	6.121
46	0x2E	**objc_edit()**	Manipulates an editable object.	6.118
47	0x2F	**objc_change()**	Changes the state of an object.	6.115
48	0x30	**objc_sysvar()**	Reads/modifies the system defaults for 3D effects.	6.121
50	0x32	**form_do()**	Manages a user-defined form.	6.81
51	0x33	**form_dial()**	Reserves/releases screen space for forms.	6.80
52	0x34	**form_alert()**	Manages a generic alert.	6.77
53	0x35	**form_error()**	Manages a generic error alert.	6.82
54	0x36	**form_center()**	Centers an object tree on screen.	6.79
55	0x37	**form_keybd()**	Provides a system-level editable field handler.	6.83
56	0x38	**form_button()**	Provides a system-level button handler.	6.78
70	0x46	**graf_rubberbox()**	Controls the shrinking/enlarging of a box outline.	6.97
71	0x47	**graf_dragbox()**	Controls the moving of a box outline.	6.91
72	0x48	**graf_movebox()**	Draws a moving box.	6.96
73	0x49	**graf_growbox()**	Draws an expanding box.	6.92

A.14 – Functions by Opcode

Dec	Hex	Function	Summary	Page
74	0x50	**graf_shrinkbox()**	Draws a shrinking box.	6.98
75	0x51	**graf_watchbox()**	Selects/draws an object depending on the position of the mouse.	6.100
76	0x52	**graf_slidebox()**	Controls a slider outline.	6.99
77	0x53	**graf_handle()**	Obtains **AES** workstation attributes.	6.92
78	0x54	**graf_mouse()**	Defines the mouse form.	6.94
79	0x55	**graf_mkstate()**	Provides information about the mouse state.	6.93
80	0x56	**scrp_read()**	Determines the system scrap directory.	6.135
81	0x57	**scrp_write()**	Sets the system scrap directory.	6.136
90	0x58	**fsel_input()**	Manages the file selector.	6.88
91	0x59	**fsel_exinput()**	Manages the extended file selector.	6.87
100	0x64	**wind_create()**	Creates a window.	6.150
101	0x65	**wind_open()**	Opens a window.	6.158
102	0x66	**wind_close()**	Closes a window.	6.150
103	0x67	**wind_delete()**	Deletes a window.	6.152
104	0x68	**wind_get()**	Returns window attributes.	6.153
105	0x69	**wind_set()**	Sets a window attribute.	6.158
106	0x6A	**wind_find()**	Determines the window at given pixel coordinates.	6.152
107	0x6B	**wind_update()**	Manages the window update semaphore.	6.161
108	0x6C	**wind_calc()**	Calculates window extents.	6.149
109	0x6D	**wind_new()**	Removes all windows.	6.157
110	0x6E	**rsrc_load()**	Loads a disk-based resource file.	6.128
111	0x6F	**rsrc_free()**	Releases a resource file from memory.	6.127
112	0x70	**rsrc_gaddr()**	Calculates the address of a resource element.	6.127
113	0x71	**rsrc_saddr()**	Sets the address of a resource element.	6.130
114	0x72	**rsrc_obfix()**	Changes the coordinates of an object from character-based to pixel-based.	6.129
115	0x73	**rsrc_rcfix()**	Changes the coordinates of a resource file from character-based to pixel-based.	6.130
120	0x78	**shel_read()**	Determine's the processes parent and command tail.	6.141
121	0x79	**shel_write()**	Manages process loading and control.	6.142
122	0x7A	**shel_get()**	Copies data from the system's shell buffer.	6.140
123	0x7B	**shel_put()**	Stores data in the system's shell buffer.	6.141
124	0x7C	**shel_find()**	Searches the **AES**'s path for a file.	6.139
125	0x7D	**shel_envrn()**	Searches the system environment string.	6.139
130	0x82	**appl_getinfo()**	Returns information about the **AES**.	6.48

VDI Functions by Opcode

Opcode, Subopcode(s) (if required)	Function	Summary	Page
N/A	**vq_gdos()**	Test for presence of **GDOS**.	7.92
-1, 6	**v_set_app_buff()**	Reserve bezier workspace.	7.77
1	**v_opnwk()**	Open physical workstation.	7.66
2	**v_clswk()**	Close a physical workstation.	7.35
3	**v_clrwk()**	Close a physical workstation.	7.34
4	**v_updwk()**	Update workstation.	7.78
5, 1	**vq_chcells()**	Return alpha screen size.	7.87
5, 2	**v_exit_cur()**	Exit text mode.	7.46
5, 3	**v_enter_cur()**	Enter text mode.	7.45
5, 4	**v_curup()**	Move text cursor up one row.	7.40
5, 5	**v_curdown()**	Move text cursor down one row.	7.37
5, 6	**v_curright()**	Move text cursor right one row.	7.38
5, 7	**v_curleft()**	Move text cursor up one row.	7.38
5, 8	**v_curhome()**	Home text cursor.	7.37
5, 9	**v_eeos()**	Erase to end of screen.	7.42
5, 10	**v_eeol()**	Erase to end of line.	7.41
5, 11	**vs_curaddress()**	Position text cursor.	7.126
5, 12	**v_curtext()**	Output text (alpha mode).	7.39
5, 13	**v_rvon()**	Reverse text on (alpha mode).	7.75
5, 14	**v_rvoff()**	Reverse text off (alpha mode).	7.75
5, 15	**vq_curaddress()**	Inquire text cursor location.	7.89
5, 16	**vq_tabstatus()**	Get availability of tablet.	7.95
5, 17	**v_hardcopy()**	Output screen to printer.	7.57
5, 18	**v_dspcur()**	Display text cursor.	7.40
5, 19	**v_rmcur()**	Remove text cursor.	7.74
5, 20	**v_form_adv()**	Advance printer page.	7.48
5, 21	**v_output_window()**	Output window of page to printer.	7.68
5, 22	**v_clear_disp_list()**	Clear display list.	7.34
5, 23	**v_bit_image()**	Render bit-image file.	7.31
5, 24	**vq_scan()**	Return printer scan heights.	7.94
5, 25	**v_alpha_text()**	Output printer text (alpha mode).	7.23
5, 60	**vs_palette()**	Set color palette.	7.127
5, 81	**vt_resolution()**	Set tablet resolution.	7.165
5, 82	**vt_axis()**	Set tablet axis resolution.	7.164
5, 83	**vt_origin()**	Set tablet origin.	7.164
5, 84	**vq_tdimensions()**	Return tablet X and Y dimensions.	7.96
5, 85	**vt_alignment()**	Set tablet alignment.	7.163
5, 91	**vqp_films()**	Return camera film types.	7.101
5, 92	**vqp_state()**	Return camera driver state.	7.101
5, 93	**vsp_state()**	Set camera driver state.	7.145
5, 94	**vsp_save()**	Save camera driver state.	7.145
5, 95	**vsp_message()**	Supress camera screen messages.	7.144
5, 96	**vqp_error()**	Return camera error status.	7.100
5, 98	**v_meta_extents()**	Specify metafile bounding box.	7.60

A.16 – Functions by Opcode

Opcode, Subopcode(s) (if required)	Function	Summary	Page
5, 99†	**v_write_meta()**	Write metafile item.	7.79
5, 99, 0†	**vm_pagesize()**	Set metafile page size.	7.85
5, 99, 1†	**vm_coords()**	Set metafile coordinate system.	7.83
5, 99, 32, 1†	**v_bez_qual()**	Set bezier quality.	7.30
5, 100	**vm_filename()**	Set metafile filename.	7.84
5, 102	**v_fontinit()**	Select a new system font.	7.48
5, 2000	**v_pgcount()**	Specify laser printer copies.	7.69
6	**v_pline()**	Draw a polyline.	7.71
6, 13	**v_bez()**	Draw a bezier curve.	7.26
7	**v_pmarker()**	Draw polymarkers.	7.72
8	**v_gtext()**	Output graphic text.	7.56
9	**v_fillarea()**	Draw a filled polygon.	7.46
9, 13	**v_bez_fill()**	Draw a filled bezier curve.	7.27
10	**v_cellarray()**	Draw a cell array.	7.32
11, 1	**v_bar()**	Draw a rectangle.	7.25
11, 2	**v_arc()**	Draw an arc.	7.24
11, 3	**v_pieslice()**	Draw a pieslice.	7.70
11, 4	**v_circle()**	Draw a circle.	7.33
11, 5	**v_ellipse()**	Draw an ellipse	7.43
11, 6	**v_ellarc()**	Draw an elliptical arc.	7.42
11, 7	**v_ellpie()**	Draw an elliptical pie segment.	7.44
11, 8	**v_rbox()**	Draw a rounded-rectangle.	7.72
11, 9	**v_rfbox()**	Draw a filled rounded-rectangle.	7.73
11, 10	**v_justified()**	Output justified text.	7.58
11, 13†	**v_bez_off()**	Disable bezier drawing.	7.28
11, 13†	**v_bez_on()**	Enable bezier drawing.	7.29
12	**vst_height()**	Set graphic text height (in pixels).	7.153
13	**vst_rotation()**	Set graphic text rotation.	7.156
14	**vs_color()**	Set color palette index.	7.126
15	**vsl_type()**	Set line type.	7.135
16	**vsl_width()**	Set line width.	7.137
17	**vsl_color()**	Set line color.	7.134
18	**vsm_type()**	Set marker type.	7.142
19	**vsm_height()**	Set marker height.	7.139
20	**vsm_color()**	Set marker color.	7.138
21	**vst_font()**	Set graphic text font.	7.152
22	**vst_color()**	Set graphic text color.	7.150
23	**vsf_interior()**	Set fill interior type.	7.129
24	**vsf_style()**	Set fill style type.	7.131
25	**vsf_color()**	Set fill color.	7.129
26	**vq_color()**	Inquire palette index.	7.88
27	**vq_cellarray()**	Inquire cell array.	7.86
28†	**vrq_locator()**	Poll for mouse/keyboard input.	7.121
28†	**vsm_locator()**	Sample mouse/keyboard input.	7.140
29†	**vrq_valuator()**	Poll for ‘valuator’ input.	7.123
29†	**vsm_valuator()**	Sample ‘valuator’ input.	7.143
30†	**vrq_choice()**	Poll for ‘choice’ input.	7.121
30†	**vsm_choice()**	Sample input from ‘choice’ device.	7.138

Opcode, Subopcode(s) (if required)	Function	Summary	Page
31†	**vrq_string()**	Poll for keyboard string input.	7.122
31†	**vsm_string()**	Sample keyboard string input.	7.141
32	**vswr_mode()**	Set writing mode.	7.162
33	**vsin_mode()**	Set input mode.	7.133
35	**vql_attributes()**	Return line attributes.	7.98
36	**vqm_attributes()**	Return marker attributes.	7.99
37	**vqf_attributes()**	Return fill area attributes.	7.96
38	**vqt_attributes()**	Return text attributes.	7.104
39	**vst_alignment()**	Set graphic text alignment.	7.146
100	**v_opnvwk()**	Open virtual workstation.	7.61
101	**v_clsvwk()**	Close a virtual workstation.	7.35
102	**vq_extnd()**	Inquire workstation attributes.	7.89
103	**v_contourfill()**	Fill an irregularly shaped region.	7.36
104	**vsf_perimeter()**	Set fill perimeter visibility.	7.130
105	**v_get_pixel()**	Read screen pixel value.	7.55
106	**vst_effects()**	Set graphic text effects.	7.150
107	**vst_point()**	Set graphic text height (by point).	7.155
108	**vsl_ends()**	Set line end style.	7.134
109	**vro_cpyfm()**	Copy raster (opaque mode).	7.119
110	**vr_trnfm()**	Transform raster form.	7.117
111	**vsc_form()**	Set mouse form.	7.128
112	**vsf_udpat()**	Set user defined fill pattern	7.132
113	**vsl_udsty()**	Set user-defined line style.	7.136
114	**vr_recfl()**	Output filled rectangle.	7.117
115	**vqin_mode()**	Return input mode for device.	7.97
116	**vqt_extent()**	Return graphic text extent.	7.107
117	**vqt_width()**	Return graphic character width.	7.115
118	**vex_timv()**	Install timer tick routine.	7.83
119	**vst_load_fonts()**	Load fonts from disk.	7.154
120	**vst_unload_fonts()**	Unload fonts.	7.160
121	**vrt_cpyfm()**	Copy raster (transparent mode).	7.124
122	**v_show_c()**	Show mouse cursor.	7.77
123	**v_hide_c()**	Hide mouse cursor.	7.57
124	**vq_mouse()**	Get mouse position and state.	7.93
125	**vex_butv()**	Install mouse button routine.	7.80
126	**vex_motv()**	Install mouse movement routine.	7.82
127	**vex_curv()**	Install mouse rendering routine.	7.81
128	**vq_key_s()**	Get shift key status.	7.93
129	**vs_clip()**	Set clipping rectangle.	7.125
130	**vqt_name()**	Return font name and index.	7.113
131	**vqt_fontinfo()**	Return font size information.	7.111
232	**vqt_fontheader()**	Copy the Speedo font header into a user defined buffer.	7.110
234	**vqt_trackkern()**	Inquire about current track kerning.	7.114
235	**vqt_pairkern()**	Inquire about current pair kerning.	7.115
236	**vst_charmap()**	Set ASCII/Speedo index interpretation mode.	7.149
237	**vst_kern()**	Set kerning modes.	7.154
239	**v_getbitmap_info()**	Return Speedo font bitmap extents.	7.53
240†	**vqt_f_extent()**	Return outline text extent.	7.108

Opcode, Subopcode(s) (if required)	Function	Summary	Page
240†	**vqt_f_extent16()**	Return 16-bit outline text extent.	7.109
241†	**v_ftext()**	Output outlined text.	7.49
241†	**v_ftext16()**	Output 16-bit outlined text.	7.50
241†	**v_ftext_offset()**	Output outlined text with individual character offsets.	7.51
241†	**v_ftext_offset16()**	Output 16-bit outlined text with individual character offsets.	7.52
242	**v_killoutline()**	Free character outline (no longer used with **SpeedoGDOS**).	7.59
243	**v_getoutline()**	Return character outline.	7.54
244	**vst_scratch()**	Set outline scratch buffer.	7.157
245	**vst_error()**	Set **GDOS** error reporting mode.	7.151
246†	**vst_arbpt()**	Set outline text point size.	7.147
246†	**vst_arbpt32()**	Set outline text point size to a fix31 value.	7.148
247	**vqt_advance()**	Return character advance vector.	7.102
247	**vqt_advance32()**	Return character advance vector as a fix31 value.	7.103
248	**vqt_devinfo()**	Return device information.	7.106
249	**v_savecache()**	Save bitmap cache to disk.	7.76
250	**v_loadcache()**	Load bitmap cache from disk.	7.59
251	**v_flushcache()**	Flush outline font cache.	7.47
252†	**vst_setsize()**	Set outline text proportion.	7.158
252†	**vst_setsize32()**	Set outline text proportion to a fix31 value.	7.159
253	**vst_skew()**	Set outline text skew factor.	7.160
254	**vqt_get_table()**	Return character mappings.	7.112
255	**vqt_cachesize()**	Return bitmap cache size	7.105

† These functions share an opcode and sub-opcode.