Contents

What is the System Information SDK ?

The **System Information SDK (Software Development Kit)** is a Dynamic Link Library and a Library of Visual Basic function calls that enables developers to determine configuration information about a computer.

The **System Information SDK** itself is shareware. `SYSINFO.DLL` may be distributed royalty free with any application. All other SDK components may not be distributed with any applications.

In other words, just like the Windows 3.1 SDK.

Components of the System Information SDK

The **System Information SDK** consists of the following components:

Component	Description
`SYSINFO.DLL`	The DLL used to determine the System Information.
`SYSINFO.BAS`	A Visual Basic module containing "wrappers" for the `SYSINFO.DLL` function calls. This file does not necessarily need to be included. It is strongly suggested that you use the function calls included in this module, as some of them perform character string conversion on information received from `SYSINFO.DLL`. Some function calls do nothing but pass information through to `SYSINFO.DLL`. For the sake of consistency, it is suggested you use the function calls included in this library. This will guarantee that you will not encounter any general protection faults. Because of the importance of using `SYSINFO.BAS`, the code examples only reference function calls to `SYSINFO.BAS`.
`SYSINFO.GBL`	A Visual Basic global file containing the `SYSINFO.DLL` declarations along with constant declarations. Please do not alter the constants or their values. They are required to be synchronized with the constants in `SYSINFO.DLL`
`SYSINFO.HLP`	This help file.
`SAMPLE.MAK`	Sample Visual Basic application make file.
`SAMPLE.FRM`	Sample Visual Basic form.
`VBSYSINF.EXE`	VB System Information 1.01. Previously released, but included here to take advantage of the additional debugging in `SYSINFO.DLL`.
`VBSYSINF.HLP`	Help file for VB System Information 1.01.
`THREED.VBX`	Custom control required by VB System Information 1.01.
`README.TXT`	Text file containing other information.

Function calls are prefaced by "SI" to indicate they are function calls located in `SYSINFO.BAS`.

This SDK was designed to interface with Visual Basic programs. `SYSINFO.DLL` can also be used from any other language that supports calls to DLLs. Information on how to link the DLL to a particular language will have to be reverse engineered from `SYSINFO.BAS`. `SYSINFO.DLL` is not officially supported in other languages than Visual Basic.

This SDK is as accurate as possible. If any inconsistencies, errors, or omissions are present in the code or documentation, please E-Mail Snowy Mountain Software with the information. It is with your assistance, that this product be made as accurate as possible.

Visual Basic Requirements

The **System Information SDK** requires Visual Basic 2.0 or higher, Windows 3.X, and a 80286 or higher processor.

The sample application included with the SDK was created using Visual Basic 3.0, but should be compatible with Visual Basic 2.0.

Registration Information

To register the **System Information SDK**, please send $10.00 to:

> **Snowy Mountain Software**
> 10968 - 126 Street
> Edmonton, Alberta, Canada
> T5M 0P5

Upon registration, you will receive the next revision of the **System Information SDK**.

Software updates and bug fixes will be posted on CompuServe.

Please send your comments, questions and bug reports to Snowy Mountain Software at CompuServe ID 72123,3402.

SIGetModelType

Description This function determines the model/manufacturer of the computer.

Syntax **INTEGER SIGetModelType** ()

Returns An integer indicating the model type of the computer.

Remark This function returns `ML_MSDOS` if no other model type can be determined. This function is as accurate as possible. If this function does not identify a computer type properly, please use the BIOS capture feature of VB System Information 1.01 (included with the SDK) & upload the results to Snowy Mountain Software on CompuServe @ ID 72123,3402.

Example

```
'The following code demonstrates determining the computer model
'type. For this example, the computer is a COMPAQ.

Dim cModelType As String
Dim nModelType As Integer

nModelType = SIGetModelType()

Select Case nModelType

    Case ML_3COM
        cModelType = "3Com"

    Case ML_ACER
        cModelType = "Acer"

    Case ML_ALR
        cModelType = "ALR"

    Case ML_AST
        cModelType = "AST"

    Case ML_ALTEC
        cModelType = "Altec"

    Case ML_ALTOS
        cModelType = "Altos"

    Case ML_AMSTRAD
        cModelType = "Amstrad"

    Case ML_APRICOT
        cModelType = "Apricot"

    Case ML_ARCHE
        cModelType = "Arche"

    Case ML_ATandT
        cModelType = "AT&T"

    Case ML_ATANDTSAFARINOTEBOOK
        cModelType = "AT&T Safari notebook"
```

```
Case ML_BLACKSHIP
    cModelType = "Blackship"

Case ML_BULL
    cModelType = "Bull"

Case ML_COMMODORE
    cModelType = "Commodore"

Case ML_COMPUADD
    cModelType = "CompuAdd"

Case ML_COLUMBIA
    cModelType = "Columbia"

Case ML_COMPAQ
    cModelType = "Compaq"

Case ML_DATAGENERAL
    cModelType = "Data General"

Case ML_DEC
    cModelType = "Digital Equip"

Case ML_DELL
    cModelType = "Dell"

Case ML_DOLCH
    cModelType = "Dolch"

Case ML_EVEREX
    cModelType = "Everex"

Case ML_EMERSON
    cModelType = "Emerson"

Case ML_EAGLE
    cModelType = "Eagle"

Case ML_EPSON
    cModelType = "Epson"

Case ML_GATEWAY
    cModelType = "Gateway"

Case ML_GOLDSTAR
    cModelType = "Goldstar"

Case ML_HAUPPAUGE
    cModelType = "Hauppauge"

Case ML_HP
    cModelType = "Hewlett-Packard"

Case ML_HEADSTART
    cModelType = "Headstart"
```

```
Case ML_HYUNDIA
    cModelType = "Hyundai"

Case ML_IBMPCXT286
    cModelType = "IBM PC/XT 286"

Case ML_IBMPCAT
     cModelType = "IBM PC/AT"

Case ML_IBMPS1M2011
     cModelType = "IBM PS/1 Model 2011"

Case ML_IBMPS1M2121
     cModelType = "IBM PS/1 Model 2121"

Case ML_IBMPS2M25
     cModelType = "IBM PS/2 Model 25"

Case ML_IBMPS2M30
     cModelType = "IBM PS/2 Model 30"

Case ML_IBMPS2M35
     cModelType = "IBM PS/2 Model 35"

Case ML_IBMPS2ML40
     cModelType = "IBM PS/2 Model L40"

Case ML_IBMPS2M50
     cModelType = "IBM PS/2 Model 50"

Case ML_IBMPS2M50Z
     cModelType = "IBM PS/2 Model 50Z"

Case ML_IBMPS2M55LS
     cModelType = "IBM PS/2 Model 55LS"

Case ML_IBMPS2M55SX
     cModelType = "IBM PS/2 Model 55SX"

Case ML_IBMPS2M57
     cModelType = "IBM PS/2 Model 57"

Case ML_IBMPS2M60
     cModelType = "IBM PS/2 Model 60"

Case ML_IBMPS2M65
     cModelType = "IBM PS/2 Model 65"

Case ML_IBMPS2M70
     cModelType = "IBM PS/2 Model 70"

Case ML_IBMPS2MP70
     cModelType = "IBM PS/2 Model P70"

Case ML_IBMPS2MP75
     cModelType = "IBM PS/2 Model P75"
```

```
Case ML_IBMPS2M80
     cModelType = "IBM PS/2 Model 80"

Case ML_IBMPS2M90
     cModelType = "IBM PS/2 Model 90"

Case ML_IBMPS2M95
     cModelType = "IBM PS/2 Model 95"

Case ML_LEADINGEDGE
    cModelType = "Leading Edge"

Case ML_MSDOS386SLAPM
    cModelType = "Intel 386SL Based System with APM"

Case ML_MSDOS
    cModelType = "MS-DOS System"

Case ML_MEMOREX
    cModelType = "Memorex"

Case ML_MITAC
    cModelType = "Mitac"

Case ML_MITSUBISHI
    cModelType = "Mitsubishi"

Case ML_MITSUBA
    cModelType = "Mitsuba"

Case ML_MATSUSHITA
    cModelType = "Matsushita"

Case ML_MICROEXPRESS
    cModelType = "Micro Express"

Case ML_MICROTELESIS
    cModelType = "Micro Telesis"

Case ML_NORTHGATE
    cModelType = "Northgate"

Case ML_NECULTRALITE
    cModelType = "NEC UltraLite"

Case ML_NCR
    cModelType = "NCR"

Case ML_NECPOWERMATE
    cModelType = "NEC PowerMate"

Case ML_NEC
    cModelType = "NEC"

Case ML_OLIVETTI
    cModelType = "Olivetti"
```

```
Case ML_PACKARDBELL
    cModelType = "Packard Bell"

Case ML_PANASONIC
    cModelType = "Panasonic"

Case ML_PCDESIGNS
    cModelType = "PC Designs"

Case ML_POLYWELL
    cModelType = "Polywell"

Case ML_PCDIRECT
    cModelType = "PC Direct"

Case ML_PCSLIMITED
    cModelType = "PC's Limited"

Case ML_PCSOURCE
    cModelType = "PC Source"

Case ML_PCBRAND
    cModelType = "PC Brand"

Case ML_PROTEUS
    cModelType = "Proteus"

Case ML_PHILIPS
    cModelType = "Philips"

Case ML_SWAN
    cModelType = "Swan"

Case ML_SIEMENS
    cModelType = "Siemens"

Case ML_SPERRY
    cModelType = "Sperry"

Case ML_SHARP
    cModelType = "Sharp"

Case ML_TANDEM
    cModelType = "Tandem"

Case ML_TI
    cModelType = "Texas Instruments"

Case ML_TRISTAR
    cModelType = "Tri-Star"

Case ML_TWINHEAD
    cModelType = "Twinhead"

Case ML_TANDY
    cModelType = "Tandy"
```

```
Case ML_TOSHIBA1000SE
    cModelType = "Toshiba 1000SE"

Case ML_TOSHIBA1000XE
    cModelType = "Toshiba 1000XE"

Case ML_TOSHIBA1000
    cModelType = "Toshiba 1000"

Case ML_TOSHIBA1200XE
    cModelType = "Toshiba 1200XE"

Case ML_TOSHIBA1200
    cModelType = "Toshiba 1200"

Case ML_TOSHIBA1600
    cModelType = "Toshiba 1600"

Case ML_TOSHIBA3100SX
    cModelType = "Toshiba 3100SX"

Case ML_TOSHIBA3100
    cModelType = "Toshiba 3100"

Case ML_TOSHIBA3200SX
    cModelType = "Toshiba 3200SX"

Case ML_TOSHIBA3200
    cModelType = "Toshiba 3200"

Case ML_TOSHIBA3300SL
    cModelType = "Toshiba 3300SL"

Case ML_TOSHIBA5100
    cModelType = "Toshiba 5100"

Case ML_TOSHIBA5200
    cModelType = "Toshiba 5200"

Case ML_TOSHIBA5300
    cModelType = "Toshiba 5300"

Case ML_TOSHIBA
    cModelType = "Toshiba"

Case ML_TANDON
    cModelType = "Tandon"

Case ML_UNISYS
    cModelType = "Unisys"

Case ML_WANG
    cModelType = "Wang"

Case ML_WYSE
    cModelType = "Wyse"
```

```
            Case ML_XEROX
                cModelType = "Xerox"

            Case ML_ZENITH
                cModelType = "Zenith"

            Case ML_ZEOS
                cModelType = "Zeos"

        End Select

        Print "Computer Model Type: " + cModelType
```

Output

```
Computer Model Type: Compaq
```

Return Codes

CODE	VALUE	MEANING
ML_3COM	1	The computer model type is 3Com
ML_ACER	2	The computer model type is Acer
ML_ALR	3	The computer model type is ALR
ML_AST	4	The computer model type is AST
ML_ALTEC	5	The computer model type is Altec
ML_ALTOS	6	The computer model type is Altos
ML_AMSTRAD	7	The computer model type is Amstrad
ML_APRICOT	8	The computer model type is Apricot
ML_ARCHE	9	The computer model type is Arche
ML_ATandT	10	The computer model type is AT & T
ML_ATANDTSAFARINOTEBOOK	11	The computer model type is AT & T Safari Notebook
ML_BLACKSHIP	12	The computer model type is Blackship
ML_BULL	13	The computer model type is Bull
ML_COMMODORE	14	The computer model type is Commodore
ML_COMPUADD	15	The computer model type is CompuAdd
ML_COLUMBIA	16	The computer model type is Columbia
ML_COMPAQ	17	The computer model type is Compaq
ML_DATAGENERAL	18	The computer model type is Data General
ML_DEC	19	The computer model type is Digital Equipment Corporation
ML_DELL	20	The computer model type is Dell
ML_DOLCH	21	The computer model type is Dolch
ML_EVEREX	22	The computer model type is Everex
ML_EMERSON	23	The computer model type is Emerson
ML_EAGLE	24	The computer model type is Eagle

ML_EPSON	25	The computer model type is Epson
ML_GATEWAY	26	The computer model type is Gateway
ML_GOLDSTAR	27	The computer model type is Goldstar
ML_HAUPPAUGE	28	The computer model type is Hauppauge
ML_HP	29	The computer model type is Hewlett-Packard
ML_HEADSTART	31	The computer model type is Headstart
ML_HYUNDIA	32	The computer model type is Hyundai
ML_IBMPCXT286	33	The computer model type is IBM PC/XT 286
ML_IBMPCAT	34	The computer model type is IBM PC/AT
ML_IBMPS1M2011	35	The computer model type is IBM PS/1 Model 2011
ML_IBMPS1M2121	36	The computer model type is IBM PS/1 Model 2121
ML_IBMPS2M25	37	The computer model type is IBM PS/2 Model 25
ML_IBMPS2M30	38	The computer model type is IBM PS/2 Model 30
ML_IBMPS2M35	39	The computer model type is IBM PS/2 Model 35
ML_IBMPS2ML40	40	The computer model type is IBM PS/2 Model L40
ML_IBMPS2M50	41	The computer model type is IBM PS/2 Model 50
ML_IBMPS2M50Z	42	The computer model type is IBM PS/2 Model 50Z
ML_IBMPS2M55LS	43	The computer model type is IBM PS/2 Model 55LS
ML_IBMPS2M55SX	44	The computer model type is IBM PS/2 Model 55SX
ML_IBMPS2M57	45	The computer model type is IBM PS/2 Model 57
ML_IBMPS2M60	46	The computer model type is IBM PS/2 Model 60
ML_IBMPS2M65	47	The computer model type is IBM PS/2 Model 65
ML_IBMPS2M70	48	The computer model type is IBM PS/2 Model 70
ML_IBMPS2MP70	49	The computer model type is IBM PS/2 Model P70
ML_IBMPS2MP75	50	The computer model type is IBM PS/2 Model P75
ML_IBMPS2M80	51	The computer model type is IBM PS/2 Model 80
ML_IBMPS2M90	52	The computer model type is IBM PS/2 Model 90
ML_IBMPS2M95	53	The computer model type is IBM PS/2 Model 95
ML_LEADINGEDGE	55	The computer model type is Leading Edge
ML_MSDOS386SLAPM	56	The computer model type is Intel 386SL Based System with APM
ML_MSDOS	57	The computer model type is MS-DOS
ML_MEMOREX	58	The computer model type is Memorex
ML_MITAC	59	The computer model type is Mitac
ML_MITSUBISHI	60	The computer model type is Mitsubishi
ML_MITSUBA	61	The computer model type is Mitsuba

ML_MATSUSHITA	62	The computer model type is Matsushita
ML_MICROEXPRESS	63	The computer model type is Micro Express
ML_MICROTELESIS	64	The computer model type is Micro Telesis
ML_NORTHGATE	65	The computer model type is Northgate
ML_NECULTRALITE	66	The computer model type is NEC UltraLite
ML_NECPOWERMATE	67	The computer model type is NCR
ML_NEC	68	The computer model type is NEC PowerMate
ML_NCR	69	The computer model type is NEC
ML_OLIVETTI	70	The computer model type is Olivetti
ML_PACKARDBELL	71	The computer model type is Packard Bell
ML_PANASONIC	72	The computer model type is Panasonic
ML_PCDESIGNS	73	The computer model type is PC Designs
ML_POLYWELL	74	The computer model type is Polywell
ML_PCDIRECT	75	The computer model type is PC Direct
ML_PCSLIMITED	76	The computer model type is PC's Limited
ML_PCSOURCE	77	The computer model type is PC Source
ML_PCBRAND	78	The computer model type is PC Brand
ML_PROTEUS	79	The computer model type is Proteus
ML_PHILIPS	80	The computer model type is Philips
ML_SWAN	81	The computer model type is Swan
ML_SIEMENS	82	The computer model type is Siemens
ML_SPERRY	83	The computer model type is Sperry
ML_SHARP	84	The computer model type is Sharp
ML_TANDEM	85	The computer model type is Tandem
ML_TI	86	The computer model type is Texas Instruments
ML_TRISTAR	87	The computer model type is Tri-Star
ML_TWINHEAD	88	The computer model type is Twinhead
ML_TANDY	89	The computer model type is Tandy
ML_TOSHIBA1000SE	90	The computer model type is Toshiba 1000SE
ML_TOSHIBA1000XE	91	The computer model type is Toshiba 1000XE
ML_TOSHIBA1000	92	The computer model type is Toshiba 1000
ML_TOSHIBA1200XE	93	The computer model type is Toshiba 1200XE
ML_TOSHIBA1200	94	The computer model type is Toshiba 1200
ML_TOSHIBA1600	95	The computer model type is Toshiba 1600
ML_TOSHIBA3100SX	96	The computer model type is Toshiba 3100SX
ML_TOSHIBA3100	97	The computer model type is Toshiba 3100

ML_TOSHIBA3200SX	98	The computer model type is Toshiba 3200SX
ML_TOSHIBA3200	99	The computer model type is Toshiba 3200
ML_TOSHIBA3300SL	100	The computer model type is Toshiba 3300SL
ML_TOSHIBA5100	101	The computer model type is Toshiba 5100
ML_TOSHIBA5200	102	The computer model type is Toshiba 5200
ML_TOSHIBA5300	103	The computer model type is Toshiba
ML_TOSHIBA	104	The computer model type is Toshiba
ML_TANDON	105	The computer model type is Tandon
ML_UNISYS	106	The computer model type is Unisys
ML_WANG	107	The computer model type is Wang
ML_WYSE	108	The computer model type is Wyse
ML_XEROX	109	The computer model type is Xerox
ML_ZENITH	110	The computer model type is Zenith
ML_ZEOS	111	The computer model type is Zeos

SIGetBIOSManufacturer

Description This function determines the manufacturer of the computer's BIOS.

Syntax **INTEGER SIGetBIOSManufacturer** (*nBIOSManufacturer* **As Integer**)

Parameter	**Description**
nBIOSManufacturer	Upon successful completion, *nBIOSManufacturer* will contain one of the following BIOS manufacturers.

CODE	VALUE	MEANING
`BT_AMI`	1	American Megatrends Inc.
`BT_PHOENIX`	2	Phoenix Technologies Ltd.
`BT_AWARD`	3	Award
`BT_COMPAQ`	4	Compaq
`BT_TOSHIBA`	5	Toshiba
`BT_ZENITH`	6	Zenith
`BT_IBM`	7	IBM

Returns An integer indicating the success or failure of the function call.

Remark This function returns `SISDK_FAIL` if the BIOS manufacturer cannot be determined.

Example

```
'The following code demonstrates determining the BIOS
'manufacturer. For this example, the BIOS manufacturer is
'Phoenix.

Dim cBIOSManufacturer As String
Dim nBIOSManufacturer As Integer
Dim nReturnCode As Integer

nReturnCode = SIGetBIOSManufacturer(nBIOSManufacturer)

Select Case nReturnCode

    Case SISDK_FAIL
        cBIOSManufacturer = "Unknown"

    Case SISDK_SUCCESS

        Select Case nBIOSManufacturer

            Case BT_AMI
                cBIOSManufacturer = "American Megatrends
                                    Inc."
            Case BT_PHOENIX
                cBIOSManufacturer = "Phoenix Technologies Ltd."

            Case BT_AWARD
                cBIOSManufacturer = "Award"

            Case BT_COMPAQ
                cBIOSManufacturer = "Compaq"

            Case BT_TOSHIBA
                cBIOSManufacturer = "Toshiba"

            Case BT_ZENITH
                cBIOSManufacturer = "Zenith"

            Case BT_IBM
                cBIOSManufacturer = "IBM"

        End Select

End Select

Print "BIOS Manufacturer: " + cBIOSManufacturer
```

Output

```
The BIOS Manufacturer: Phoenix Technologies Ltd.
```

Return Codes

CODE	VALUE	MEANING
SISDK_FAIL	-1	The BIOS Manufacturer is unknown.
SISDK_SUCCESS	0	The function completed successfully.

SIGetBIOSDate

Description This function determines the date of the computer's BIOS.

Syntax **INTEGER SIGetBIOSDate** (*cBIOSDate* **As String**)

Parameter	**Description**
cBIOSDate	Upon successful completion, *cBIOSDate* will contain the BIOS date.

Returns An integer indicating the success or failure of the function call.

Remark This function returns `SISDK_FAIL` and *cBIOSDate* will be a null string if the BIOS date cannot be determined.

Example

```
'The following code demonstrates determining the BIOS date.
'For this example, the BIOS date is 06/06/92.

Dim cBIOSDate As String
Dim nReturnCode As Integer

nReturnCode = SIGetBIOSDate(cBIOSDate)

Select Case nReturnCode

    Case SISDK_FAIL
        cBIOSDate = "Unknown BIOS Date"

    Case SISDK_SUCCESS
        'cBIOSDate is fine just the way it is.

End Select

Print "BIOS Date: " + cBIOSDate
```

Output

```
BIOS Date : 06/06/92
```

Return Codes

CODE	**VALUE**	**MEANING**
`SISDK_FAIL`	`-1`	The BIOS date was not found.
`SISDK_SUCCESS`	`0`	The function completed successfully.

SIGetBIOSHighArea

Description This function returns a string containing the computer's BIOS code from F000:E000 to F000:FFFF. Mainly used to capture the BIOS' area for analysis. This function call is used by the BIOS Capture function of the sample application.

Syntax **STRING SIGetBIOSHighArea** ()

Returns A string of maximum length 8192 characters containing BIOS code from F000:E000 to F000:FFFF. This string will contain only the characters, numbers, and punctuation symbols of the BIOS area. It will not contain any non-printable characters.

Example

```
'The following code demonstrates retrieving the BIOS high area.

Dim cBIOSHighArea As String

cBIOSHighArea = SIGetBIOSHighArea()
```

SIGetBIOSLowArea

Description This function returns a string containing the computer's BIOS code from F000:0000 to F000:1FFF. Mainly used to capture the BIOS' area for analysis. This function call is used by the BIOS Capture function of the sample application.

Syntax **STRING SIGetBIOSLowArea** ()

Returns A string of maximum length 2048 characters containing BIOS code from F000:0000 to F000:1FFF. This string will contain only the characters, numbers, and punctuation symbols of the BIOS area. It will not contain any non-printable characters.

Example

```
'The following code demonstrates retrieving the BIOS low area.

Dim cBIOSLowArea As String

cBIOSLowArea = SIGetBIOSLowArea()
```

SIGetMemorySizeKB

Description This function determines the amount of memory installed in a computer.

Syntax **INTEGER SIGetMemorySizeKB()**

Returns An integer indicating the amount of memory installed in a computer.

Remark If the computer only has 640K with a memory board populated with more memory, the 640K will be counted as 1MB towards the total amount of memory.

Example

```
'The following code demonstrates determining the amount of
'memory installed in a computer. For this example, the amount
'of memory installed in a computer is 8192 KB.

Print "Computer memory: " +
      Trim$(Str$(SIGetMemorySizeKB())) + " KB"
```

Output

```
Computer Memory: 8192 KB
```

SIGetDOSVersion

Description This function determines the version of DOS installed on a computer.

Syntax **SINGLE SIGetDOSVersion**()

Returns A single precision floating point number indicating the version of DOS installed on a computer. The number returned will be accurate to one decimal place.

Remark It is unknown what version of DOS will be displayed when running under a OS/2 2.x DOS box. DOS versions 4.01 & 4.02 will be identified as 4.0

Example

```
'The following code demonstrates determining the version of
'DOS installed on a computer.  For this example, the version of
'DOS is 3.3

Print "DOS version: " + Trim$(Format$((SIGetDOSVersion()),
                                    "0.0"))
```

Output

```
DOS version: 3.3
```

SIGetBusType

Description This function determines the bus type of a computer.

Syntax **INTEGER SIGetBusType**()

Returns An integer indicating the bus type of a computer.

Example

```
'The following code demonstrates determining the bus type of
'a computer. For this example, the BUS type is Microchannel.

Dim cBusType As String
Dim nBusType As Integer

nBusType = SIGetBusType()

Select Case nBusType

    Case BUS_MCA
        cBusType = "Microchannel"

    Case BUS_ISA
        cBusType = "ISA"

    Case BUS_EISA
        cBusType = "EISA"

End Select

Print "Bus type: " + cBusType
```

Output

```
Bus type: Microchannel
```

Return Codes

CODE	VALUE	MEANING
BUS_MCA	1	The bus type is Microchannel
BUS_ISA	2	The bus type is Industry Standard Architecture
BUS_EISA	3	The bus type is Extended Industry Standard Architecture

SIGetCPUType

Description This function determines CPU installed in the computer.

Syntax **INTEGER SIGetCPUType**()

Returns An integer indicating the CPU type of a computer.

Remark Future versions of this SDK will recognize SX vs. DX CPUs and eventually, the Pentium processor. Currently, if the actual CPU Type is a Pentium, the value returned will be a 80486.

Example

```
'The following code demonstrates determining the CPU type of
'a computer. For this example, the CPU type is 80386.

Dim cCPUType As String
Dim nCPUType As Integer

nCPUType = SIGetCPUType()

Select Case nCPUType

    Case CPU_80286
        cCPUType = "80286"

    Case CPU_80386
        cCPUType = "80386"

    Case CPU_80486
        cCPUType = "80486"

End Select

Print "CPU type: " + cCPUType
```

Output

```
CPU type: 80386
```

Return Codes

CODE	VALUE	MEANING
CPU_80286	2	The CPU type is 80286
CPU_80386	3	The CPU type is 80386
CPU_80486	4	The CPU type is 80486

SIGetCPUMHz

Description This function determines CPU MHz of a computer.

Syntax **INTEGER SIGetCPUMHz**()

Returns An integer indicating the CPU MHz of a computer.

Remark If the CPU chip is a Pentium, the value to be displayed is unpredictable. If the CPU speed of the computer is non-standard i.e. 28 MHz, the speed returned will be the closest value to the values listed below in **Returned Values**

Example

```
'The following code demonstrates determining the CPU MHz of a
'computer. For this example, the CPU speed is 40 MHz.

Print "CPU MHz: " + Trim$(Str$(SIGetCPUMHz())) + " MHz"
```

Output

```
CPU MHz: 40 MHz
```

Return Values

This function returns the following CPU MHz.

- 8
- 10
- 12
- 16
- 20
- 25
- 40
- 50
- 60
- 66

SIGetCoProcessorType

Description This function determines the presence of a co-processor in a computer.

Syntax **INTEGER SIGetCoProcessorType**()

Returns An integer indicating the presence of a co-processor. If a co-processor is present the co-processor type is returned.

Remark If `CPR_BUILTIN` is displayed, the co-processor is integrated with the CPU. Examples of this are the 80486 chip, the Overdrive chip, and the Pentium.

Example

```
'The following code demonstrates determining the presence of
'a co-processor in a computer. For this example, the CPU is an
'80286 and the computer has a 80287 installed.

Dim cCoProcessorType As String
Dim nCoProcessorType As Integer

nCoProcessorType = SIGetCoProcessorType()

Select Case nCoProcessorType

    Case CPR_BUILTIN
        cCoProcessorType = "Built In"

    Case CPR_NOCOPRO
        cCoProcessorType = "No co-processor"

    Case CPR_80287
        cCoProcessorType = "80287"

    Case CPR_80387
        cCoProcessorType = "80387"

End Select

Print "Co-Processor installed: " + cCoProcessorType
```

Output

```
Co-Processor installed: 80287
```

Return Codes

CODE	VALUE	MEANING
CPR_BUILTIN	-2	The co-processor is built in to the CPU
CPR_NOCOPRO	-1	A co-processor is not present
CPR_80287	1	The co-processor is an 80287
CPR_80387	2	The co-processor is an 80387

SIGetVideoCardDriverType

Description This function determines the Windows video card driver type.

Syntax **STRING SIGetVideoCardDriverType()**

Returns A string indicating the video card driver type.

Remark The string returned is the same information that is in the **display = XXXXXXXXXXX** entry in the SYSTEM.INI file.

Example

```
'The following code demonstrates determining the Windows
'video card driver type of a computer.  For this example, the
'computer has a VESA compatible video card with the Windows
'Super VGA (800x600, 16 colors) driver installed.

Print "Video card driver type: " + SIGetVideoCardDriverType()
```

Output

```
Video card driver type: Super VGA (800x600, 16 colors)
```

SIGetVideoCardDriverVersion

Description This function determines the Windows video card driver version.

Syntax **SINGLE SIGetVideoCardDriverVersion**()

Returns A single precision floating point number indicating the Windows video card driver version. The number returned will be accurate to two decimal place.

Remark The number returned when using a video card driver shipped with Windows 3.10 will be "3.10". Other video card manufacturer's may display higher numbers.

Example

```
'The following code demonstrates determining the Windows
'video card driver version of a computer.  For this example,
'the computer has a VESA compatible video card with the Windows
'3.10 Super VGA (800x600, 16 colors) driver installed.

Print "Video card driver version: " +
 Trim$(Format$(SIGetVideoCardDriverVersion(), "0.00"))
```

Output

```
Video card driver version: 3.10
```

SIGetVideoCardPixelResolution

Description This function determines the Windows video card driver pixel resolution.

Syntax **STRING SIGetVideoCardPixelResolution**()

Returns A string indicating the Windows video card driver pixel resolution. The string will consist of an integer followed by a "x" followed by another integer. See **Example**.

Example

```
'The following code demonstrates determining the Windows
'video card driver pixel resolution.  For this example, the
'computer has a VESA compatible video card with the Windows
'3.10 Super VGA (800x600, 16 colors) driver installed.

Print "Video card pixel resolution: " +
             SIGetVideoCardPixelResolution()
```

Output

```
Video card pixel resolution: 800 x 600
```

Return Strings

The following are the most common strings returned , although other values may be returned:

- 640 x 480
- 800 x 600
- 1024 x 768
- 1280 x 1024

SIGetNumVideoCardColors

Description This function determines the number of colors the Windows video card currently supports.

Syntax **LONG SIGetNumVideoCardColors**()

Returns A long (32-bit) integer indicating the number of colors the Windows video card currently supports.

Remark When a driver is currently supporting 16 million colors (24 bit color), the value returned will be "0". Due to the small number of adapters actively using this color depth, it was not a priority to support it in this release. 24 bit color depth will be supported in the next release.

Example

```
'The following code demonstrates determining the number of 'colors
the Windows video card driver currently supports. For
'this example, the computer has a VESA compatible video card
'with the Windows 3.10 Super VGA (800x600, 256 colors) driver
'installed.

Print "Number of video card colors: " +
            Trim$(Str$(SIGetNumVideoCardColors()))
```

Output

```
Number of video card colors: 256
```

Return Values

The following are the most common values returned , although other values may be returned:

- 16
- 256
- 32768
- 65536

SIGetNetBIOSComputerName

Description This function determines the unique computer name is used to identify the computer to a network operating system.

Syntax **INTEGER SIGetNetBIOSComputerName(***cNetBIOSComputerName* **As Integer)**

Parameter	**Description**
cNetBIOSComputerName	Upon successful completion *cNetBIOSComputerName* will contain the NetBIOS computer name. The string will contain up to 16 characters.

Returns An integer indicating the success or failure of the function call.

Remark A NetBIOS driver must be installed on the computer for this function to return a value. If a NetBIOS driver is not installed (i.e. NetWare), a null string will be returned. This function call is network independent and will work with any network operating system that is NetBIOS based or supports NetBIOS function calls. If the function returns `NET_NONETWORK` *cNetBIOSComputerName* will be a null string ("").

Example

```
'The following code demonstrates determining the network
'computer name. For this example, the network computer name is
'"SNOWYMOUNTAIN1"

Dim cNetBIOSComputerName As String
Dim nReturnCode As Integer

nReturnCode = SIGetNetBIOSComputerName(cNetBIOSComputerName)

Select Case nReturnCode

    Case NET_NONETWORK
        cNetBIOSComputerName = "Network Is Not Started"

    Case SISDK_SUCCESS
        'cNetBIOSComputerName is fine just the way it is.

End Select

Print "NetBIOS computer name: " + cNetBIOSComputerName
```

Output

```
NetBIOS computer name: SNOWYMOUNTAIN1
```

Return Codes

CODE	**VALUE**	**MEANING**
`NET_NONETWORK`	`-2`	A NetBIOS compatible network is not started.
`SISDK_SUCCESS`	`0`	Function returned successfully.

SIGetNetworkType

Description This function determines the brand of the network operating system

Syntax **INTEGER SIGetNetworkType**(*cNetworkType* **As Integer**)

Parameter	**Description**
cNetworkType	Upon successful completion, *cNetworkType* will contain one of the following network types.

CODE	**VALUE**	**MEANING**
`NW_NONETWORK`	`-2`	There is no network installed
`NW_3PLUSOPEN`	`1`	The network operating system is 3Com 3+Open
`NW_3PLUSSHARE`	`2`	The network operating system is 3Com 3+Share
`NW_LANTASTIC`	`3`	The network operating system is Artisoft LANtastic
`NW_BANYANVINES`	`4`	The network operating system is Banyan Vines
`NW_IBMLANSERVER`	`5`	The network operating system is IBM OS/2 LAN Server
`NW_IBMPCLAN`	`6`	The network operating system is IBM PC LAN Program
`NW_MSLANMANAGER`	`7`	The network operating system is Microsoft LAN Manager
`NW_MSNETWORK`	`8`	The network operating system is Microsoft Network (or 100% compatible)
`NW_NETWARE`	`9`	The network operating system is Novell NetWare
`NW_DECPATHWORKS`	`10`	The network operating system is DEC PATHWORKS
`NW_TCS10NET`	`11`	The network operating system is TCS 10Net

Returns An integer indicating the success or failure of the function call.

Remark If the computer is connected to a network but is not one of the listed types, `SISDK_FAIL` is returned.

If the computer is not connected to a network, this function returns `NW_NONETWORK`.

Example

```
'The following code demonstrates determining the brand of
'network operating system. For this example, the network
'operating system is LANTASTIC.

Dim cNetworkType As String
Dim nNetworkType As Integer
Dim nReturnCode As Integer

nReturnCode = SIGetNetworkType(nNetworkType)

Select Case nReturnCode

    Case SISDK_FAIL
        cNetworkType = "Unknown Network Type"

    Case SISDK_SUCCESS

        Select Case nNetworkType

            Case NW_NONETWORK
                cNetworkType = "No Network Installed"

            Case NW_3PLUSOPEN
                cNetworkType = "3Com 3+Open"

            Case NW_3PLUSSHARE
                cNetworkType = "3Com 3+Share"

            Case NW_LANTASTIC
                cNetworkType = "Artisoft LANtastic"

            Case NW_BANYANVINES
                cNetworkType = "Banyan Vines"

            Case NW_IBMLANSERVER
                cNetworkType = "IBM OS/2 LAN Server"

            Case NW_IBMPCLAN
                cNetworkType = "IBM PC LAN Program"

            Case NW_MSLANMANAGER
                cNetworkType = "Microsoft LAN Manager"

            Case NW_MSNETWORK
                cNetworkType = "Microsoft Network (or 100%
                              compatible)"

            Case NW_NETWARE
                cNetworkType = "Novell NetWare"

            Case NW_DECPATHWORKS
                cNetworkType = "DEC PATHWORKS"

            Case NW_TCS10NET
```

```
                  cNetworkType = "TCS 10Net"

          End Select

  End Select

  Print "Network type: " + cNetworkType
```

Output

```
Network type: Artisoft LANtastic
```

Return Codes

CODE	VALUE	MEANING
SISDK_FAIL	-1	The network type is unknown.
SISDK_SUCCESS	0	The function completed successfully.

SIGetDiskDriveType

Description This function determines disk drive type of a given drive.

Syntax **INTEGER SIGetDiskDriveType**(*nDriveNumber* **As Integer,** ***nDiskDriveType*** **As Integer**)

Parameter	Description
nDriveNumber	An integer indicating the drive letter to be inquired about. One of the following:
nDiskDriveType	Upon successful completion, ***nDiskDriveType*** will contain one of the following disk drive types.

INTEGER	DRIVE
1	A:
2	B:
3	C:
|	|
26	Z:

CODE	VALUE	MEANING
`DT_FLOPPY`	2	The disk drive type is a **Floppy Drive**.
`DT_LOCALHD`	3	The disk drive type is a **Local Hard Drive**.
`DT_NETWORKHD`	4	The disk drive type is a **Network Hard Drive**.

Returns An integer indicating the success or failure of the function call.

Example

```
'The following code demonstrates determining the type of disk
'drive.  For this example, the disk drive is F: and is a
'network drive.

Dim nDiskDriveType As Integer
Dim cDiskDriveType As String
Dim nDiskDriveNumber As Integer
Dim nReturnCode As Integer

nDiskDriveNumber = 6   'F:

nReturnCode = SIGetDiskDriveType(nDiskDriveNumber,
                                    nDiskDriveType)
```

```
Select Case nReturnCode

    Case DT_INVALIDPARAMETER
   cDiskDriveType = "Invalid parameter"

    Case DT_NOTEXIST
   cDiskDriveType = "A disk drive is not associated with
                        the parameter"

    Case SISDK_SUCCESS

        Select Case nDiskDriveType

            Case DT_FLOPPY
         cDiskDriveType = "Floppy drive"

             Case DT_LOCALHD
         cDiskDriveType = "Local Hard Drive"

             Case DT_NETWORKHD
          cDiskDriveType = "Network Hard Drive"

  End Select

End Select

Print "F: disk drive type: " + cDiskDriveType
```

Output

```
F: disk drive type: Network Hard Drive
```

Return Codes

CODE	VALUE	MEANING
`DT_INVALIDPARAMETER`	`-3`	The *nDriveNumber* parameter is not between 1 and 26.
`DT_NOTEXIST`	`-2`	The *nDriveNumber* is not assigned to disk drive. i.e. Trying to get the type of "B:" in a computer with one floppy drive (A:) and one hard drive (C:).
`SISDK_SUCCESS`	`0`	The function completed successfully.

SIGetLASTDRIVE

Description This function determines the LASTDRIVE environment variable of a computer.

Syntax **INTEGER SIGetLASTDRIVE**()

Returns An integer indicating the last assignable driver letter available to the computer.

Example

```
'The following code demonstrates determining the LASTDRIVE of
'a computer. For this example, the LASTDRIVE is E:

Print "LASTDRIVE: " + Trim$(Chr$(SIGetLASTDRIVE() + 64)) + ":"
```

Output

```
LASTDRIVE: E:
```

Return Codes

CODE	VALUE	MEANING
	1	The LASTDRIVE is A:
	2	The LASTDRIVE is B:
	3	The LASTDRIVE is C:
	|	|
	26	The LASTDRIVE is Z:

SIGetHDBytesPerSector

Description This function determines bytes per sector of a hard drive.

Syntax **INTEGER SIGetHDBytesPerSector**(*nHardDriveNumber* **As Integer,** *nHDBytesPerSector* **As Integer**)

Parameter	**Description**
nHardDriveNumber	An integer indicating the hard drive letter to be inquired about. One of the following:
nHDBytesPerSector	Upon successful completion, *nHDBytesPerSector* will contain the number of bytes per sector on the hard disk.

INTEGER	**DRIVE**
1	A:
2	B:
3	C:
|	|
26	Z:

Returns An integer indicating the success or failure of the function call.

Example

```
'The following code demonstrates determining the bytes per
'sector of a hard drive.  For this example, the hard drive is
'F: and the bytes per sector is 1024.

Dim nHDBytesPerSector As Integer
Dim nHardDiskNumber As Integer
Dim cHDBytesPerSector As String
Dim nReturnCode As Integer

nHardDiskNumber = 6   'F:

nReturnCode= SIGetHDBytesPerSector(nHardDiskNumber,
                                    nHDBytesPerSector)

Select Case nReturnCode

    Case DT_INVALIDPARAMETER
   cHDBytesPerSector = "Invalid parameter"

    Case DT_NOTEXIST
   cHDBytesPerSector = "A disk drive is not associated
                              with the parameter"

    Case SISDK_SUCCESS
   cHDBytesPerSector = Trim$(Str$(nHDBytesPerSector))

End Select

Print "F: drive bytes per sector: " + cHDBytesPerSector
```

Output

```
F: drive bytes per sector: 1024
```

Return Codes

CODE	VALUE	MEANING
`HD_INVALIDPARAMETER`	`-3`	The *nHardDriveNumber* parameter is not between 1 and 26.
`HD_NOTHARDDRIVE`	`-2`	The *nHardDriveNumber* is not a local hard drive. i.e. Trying to information of a floppy drive, a network drive, or the parameter passed is does not reference a valid drive letter ("B:" in a computer with one floppy drive (A:) and one hard drive (C:)).
`SISDK_SUCCESS`	`0`	The function completed successfully.

SIGetHDHeads

Description This function determines the number of heads of a hard drive.

Syntax **INTEGER SIGetHDHeads**(*nHardDriveNumber* **As Integer,** ***nHDHeads*** **As Integer**)

Parameter	**Description**
nHardDriveNumber	An integer indicating the hard drive letter to be inquired about. One of the following:
nHDHeads	Upon successful completion, ***nHDHeads*** will contain the number of heads on the hard drive.

INTEGER	**DRIVE**
1	A:
2	B:
3	C:
|	|
26	Z:

Returns An integer indicating the success or failure of the function call.

Example

```
'The following code demonstrates determining the number of
'heads of a hard drive.  For this example, the hard drive is
'F: and the number of heads is 15.

Dim nHDHeads As Integer
Dim nHardDiskNumber As Integer
Dim cHDHeads As String
Dim nReturnCode As Integer

nHardDiskNumber = 6   'F:
nReturnCode= SIGetHDHeads(nHardDiskNumber, nHDHeads)

Select Case nReturnCode

    Case DT_INVALIDPARAMETER
   cHDHeads = "Invalid parameter"

    Case DT_NOTEXIST
   cHDHeads = "A disk drive is not associated
                              with the parameter"

    Case SISDK_SUCCESS
   cHDHeads = Trim$(Str$(nHDHeads))

End Select
```

```
            Print "F: drive number of heads: " + cHDHeads
```

Output

```
F: drive number of heads: 15
```

Return Codes

CODE	VALUE	MEANING
`HD_INVALIDPARAMETER`	`-3`	The *nHardDriveNumber* parameter is not between 1 and 26.
`HD_NOTHARDDRIVE`	`-2`	The *nHardDriveNumber* is not a local hard drive. i.e. Trying to information of a floppy drive, a network drive, or the parameter passed is does not reference a valid drive letter ("B:" in a computer with one floppy drive (A:) and one hard drive (C:)).
`SISDK_SUCCESS`	`0`	The function completed successfully.

SIGetHDSectorsPerTrack

Description This function determines the number sectors per track of a hard drive.

Syntax **INTEGER SIGetHDSectorsPerTrack**(*nHardDriveNumber* **As Integer,** *nHDSectorsPerTrack* **As Integer**)

Parameter	**Description**
nHardDriveNumber	An integer indicating the hard drive letter to be inquired about. One of the following:

INTEGER	DRIVE
1	A:
2	B:
3	C:
|	|
26	Z:

nHDSectorsPerTrack	Upon successful completion, *nHDSectorsPerTrack* will contain the number of sectors per track of the hard drive.

Returns An integer indicating the success or failure of the function call.

Example

```
'The following code demonstrates determining the sectors per
'track of a hard drive.  For this example, the hard drive is
'F: and the sectors per track is 17.

Dim nHDSectorsPerTrack As Integer
Dim nHardDiskNumber As Integer
Dim cHDSectorsPerTrack As String
Dim nReturnCode As Integer

nHardDiskNumber = 6   'F:
nReturnCode= SIGetHDSectorsPerTrack(nHardDiskNumber,
                                     nHDSectorsPerTrack)

Select Case nReturnCode

    Case DT_INVALIDPARAMETER
   cHDSectorsPerTrack = "Invalid parameter"

    Case DT_NOTEXIST
   cHDSectorsPerTrack = "A disk drive is not associated
                              with the parameter"

    Case SISDK_SUCCESS
   cHDSectorsPerTrack = Trim$(Str$(nHDSectorsPerTrack))
```

```
End Select

Print "F: drive sectors per track: " + cHDSectorsPerTrack
```

Output

```
F: drive sectors per track: 17
```

Return Codes

CODE	VALUE	MEANING
`HD_INVALIDPARAMETER`	`-3`	The *nHardDriveNumber* parameter is not between 1 and 26.
`HD_NOTHARDDRIVE`	`-2`	The *nHardDriveNumber* is not a local hard drive. i.e. Trying to information of a floppy drive, a network drive, or the parameter passed is does not reference a valid drive letter ("B:" in a computer with one floppy drive (A:) and one hard drive (C:)).
`SISDK_SUCCESS`	`0`	The function completed successfully.

SIGetHDTracks

Description This function determines the number of tracks of a hard drive.

Syntax **INTEGER SIGetHDTracks**(*nHardDriveNumber* **As Integer**)

Parameter	**Description**
nHardDriveNumber	An integer indicating the hard drive letter to be inquired about. One of the following:
nHDTracks	Upon successful completion, *nHDTracks* will contain the number of tracks of the hard drive.

INTEGER	**DRIVE**
1	A:
2	B:
3	C:
|	|
26	Z:

Returns An integer indicating the success or failure of the function call.

Example

```
'The following code demonstrates determining the number of
'tracks of a hard drive.  For this example, the hard drive is
'F: and the number of tracks is 916.

Dim nHDTracks As Integer
Dim nHardDiskNumber As Integer
Dim cHDTracks As String
Dim nReturnCode As Integer

nHardDiskNumber = 6   'F:
nReturnCode= SIGetHDTracks(nHardDiskNumber, nHDTracks)

Select Case nReturnCode

    Case DT_INVALIDPARAMETER
   cHDTracks = "Invalid parameter"

    Case DT_NOTEXIST
   cHDTracks = "A disk drive is not associated
                              with the parameter"

    Case SISDK_SUCCESS
   cHDTracks = Trim$(Str$(nHDTracks))

End Select
```

```
Print "F: drive number of tracks: " + cHDTracks
```

Output

```
F: drive number of tracks: 916
```

Return Codes

CODE	VALUE	MEANING
`HD_INVALIDPARAMETER`	-3	The *nHardDriveNumber* parameter is not between 1 and 26.
`HD_NOTHARDDRIVE`	-2	The *nHardDriveNumber* is not a local hard drive. i.e. Trying to information of a floppy drive, a network drive, or the parameter passed is does not reference a valid drive letter ("B:" in a computer with one floppy drive (A:) and one hard drive (C:)).
`SISDK_SUCCESS`	0	The function completed successfully.

SIGetTotalDiskSpaceKB

Description This function determines the total space on a disk drive.

Syntax **INTEGER SIGetTotalDiskSpaceKB(***nDriveNumber* **As Integer,** *lTotalDiskSpace* **As Long**)

Parameter	**Description**
nDriveNumber	An integer indicating the disk drive letter to be inquired about. One of the following:

INTEGER	**DRIVE**
1	A:
2	B:
3	C:
|	|
26	Z:

lTotalDiskSpace	Upon successful completion, *lTotalDiskSpace* will contain total disk space of the drive in KB.

Returns An integer indicating the success or failure of the function call.

Remark This function works on both floppy drives, hard drives, & network hard drives. For this function to work on a floppy drive, a disk must be inserted into the floppy drive.

Example

```
'The following code demonstrates determining the total space
'of a disk drive. For this example, the disk drive is C: and
'the total disk space is 120 MB.

Dim lTotalSpace As Long
Dim nDiskNumber As Integer
Dim cTotalSpace As String
Dim nReturnCode As Integer

nDiskNumber = 3   'C:

nReturnCode = SIGetTotalDiskSpaceKB(nDiskNumber,lTotalSpace)

Select Case nReturnCode

    Case DS_INVALIDPARAMETER
        cTotalSpace = "Invalid parameter"

    Case SISDK_FAIL
  cTotalSpace =  "Disk Drive not valid or ready."

    Case SISDK_SUCCESS
```

```
        cTotalSpace =  Trim$(Str$(lTotalSpace)) + " KB"

End Select

Print "C: drive total disk space: " + cTotalSpace
```

Output

```
C: drive total disk space: 122880 KB
```

Return Codes

CODE	VALUE	MEANING
`DS_INVALIDPARAMETER`	-2	The *nDriveNumber* parameter is not between 1 and 26.
`SISDK_FAIL`	-1	The disk drive is not valid or is not ready.
`SISDK_SUCCESS`	0	The function completed successfully.

SIGetFreeDiskSpaceKB

Description This function determines the free space on a disk drive.

Syntax **INTEGER SIGetFreeDiskSpaceKB(***nDriveNumber* **As Integer,** *lFreeDiskSpace* **As Long)**

Parameter	**Description**
nDriveNumber	An integer indicating the disk drive letter to be inquired about. One of the following:
lFreeDiskSpace	Upon successful completion, *lFreeDiskSpace* will contain free disk space of the drive in KB.

INTEGER	**DRIVE**
1	A:
2	B:
3	C:
|	|
26	Z:

Returns An integer indicating the success or failure of the function call.

Remark This function works on both floppy drives, hard drives, & network hard drives. For this function to work on a floppy drive, a disk must be inserted into the floppy drive.

Example

```
'The following code demonstrates determining the free space
'of a disk drive. For this example, the disk drive is C: and
'the free disk space is 16 MB.

Dim lFreeSpace As Long
Dim nDiskNumber As Integer
Dim cFreeSpace As String
Dim nReturnCode As Integer

nDiskNumber = 3   'C:

nReturnCode = SIGetFreeDiskSpaceKB(nDiskNumber,lFreeSpace)

Select Case nReturnCode

    Case DS_INVALIDPARAMETER
        cFreeSpace = "Invalid parameter"

    Case SISDK_FAIL
  cFreeSpace =  "Disk Drive not valid or ready."

    Case SISDK_SUCCESS
```

```
        cFreeSpace =  Trim$(Str$(lFreeSpace)) + " KB"
End Select
Print "C: drive free disk space: " + cFreeSpace
```

Output

```
C: drive free disk space: 16384 KB
```

Return Codes

CODE	VALUE	MEANING
DS_INVALIDPARAMETER	-2	The *nDriveNumber* parameter is not between 1 and 26.
SISDK_FAIL	-1	The disk drive is not valid or is not ready.
SISDK_SUCCESS	0	The function completed successfully.

SIGetFloppyDriveType

Description This function determines the type of floppy disk drive.

Syntax **INTEGER SIGetFloppyDriveType**(*nFloppyDriveNumber* **As Integer,** *nFloppyDriveType* **As Integer**)

Parameter	**Description**
nFloppyDriveNumber	An integer indicating the disk drive letter to be inquired about. One of the following:

INTEGER	**DRIVE**
1	A:
2	B:
3	C:
4	D:

Parameter	**Description**
nFloppyDriveType	Upon successful completion, *nFloppyDriveType* will contain one of the following floppy drive types.

CODE	**VALUE**	**MEANING**
`FD_360K`	1	360K 5 1/4 inch drive
`FD_12MB`	2	1.2MB 5 1/4 inch drive
`FD_720K`	3	720K 3 1/2 inch drive
`FD_144MB`	4	1.44MB 3 1/2 inch drive
`FD_288MB`	6	2.88MB 3 1/2 inch drive

Returns An integer indicating the success or failure of the function call.

Remark This function supports the new 2.88 MB floppy drives.

Example

```
'The following code demonstrates determining the floppy drive
'type. For this example, the disk drive is B: and the floppy
'drive type is 3½ inch 2.88 MB capacity.

Dim nFloppyDriveType As Integer
Dim cFloppyDriveType As String
Dim nReturnCode As Integer
Dim nFloppyDriveNumber As Integer

nFloppyDriveNumber = 2   'B:

nReturnCode = SIGetFloppyDriveType(nFloppyDriveNumber,
                                    nFloppyDriveType)

Select Case nReturnCode

    Case FD_INVALIDPARAMETER
   cFloppyDriveType = "Invalid Parameter"

    Case FD_NOTFLOPPYDRIVE
   cFloppyDriveType = "Floppy Drive Does Not Exist"

    Case SISDK_SUCCESS

   Select Case nFloppyDriveType

      Case FD_360K
          cFloppyDriveType = "360 KB  5 1/4 Floppy Drive"

      Case FD_12MB
          cFloppyDriveType = "1.2 MB  3 1/2 Floppy Drive"

      Case FD_720K
          cFloppyDriveType = "720 KB  5 1/4 Floppy Drive"

      Case FD_144MB
          cFloppyDriveType= "1.44 MB  3 1/4 Floppy Drive"

      Case FD_288MB
          cFloppyDriveType= "2.88 MB  3 1/4 Floppy Drive"

End Select

End Select

Print "B: drive floppy type: " + cFloppyType
```

Output

```
B: drive floppy type: 2.88 MB  3 1/4 Floppy Drive
```

Return Codes

CODE	VALUE	MEANING
`FD_INVALIDPARAMETER`	-3	The *nFloppyDriveNumber* parameter is not between 1 and 4.
`FD_NOTFLOPPYDRIVE`	-2	The *nFloppyDriveNumber* parameter is not a floppy drive (may be a hard drive though). See **SIGetDiskDriveType**
`SISDK_SUCCESS`	0	The function completed successfully.

SIGetNumFloppyDrives

Description This function retrieves the number of floppy drives installed in a computer.

Syntax **INTEGER SIGetNumFloppyDrives**()

Returns An integer indicating the number of floppy drives installed.

Example

```
'The following code demonstrates determining the number of
'floppy drives installed. For this example, the number of
'drives installed is 2.

Print "Number of floppy drives: " +
           Trim$(Str$(SIGetNumFloppyDrives()))
```

Output

```
2
```

Return Values

The following are the number of floppy drives this function will return.

- 0
- 1
- 2
- 3
- 4

SIGetMouseInterface

Description This function determines the mouse interface.

Syntax **INTEGER SIGetMouseInterface(***nMouseInterface* **As Integer)**

Parameter	Description
nMouseInterface	Upon successful completion, *nMouseInterface* will contain one of the following interface types.

CODE	VALUE	MEANING
`MI_BUS`	1	Bus
`MI_SERIAL`	2	Serial Port
`MI_INPORT`	3	InPort
`MI_PS2`	4	PS/2
`MI_HP`	5	Hewlett Packard

Returns An integer indicating the success or failure of the function call.

Remark The successful completion of this function requires a DOS mouse driver be loaded prior to starting Windows, otherwise, `MI_NODOSMOUSEDRIVER` will be returned.

NOTE: There have been unconfirmed reports that loading a DOS mouse driver high will cause this function to behave erratically. This will be fixed in the next release.

Example

```
'The following code demonstrates determining the mouse
'interface. For this example, the mouse interface is InPort

Dim cMouseInterface As String
Dim nMouseInterface As Integer
Dim nReturnCode As Integer

nReturnCode = SIGetMouseInterface(nMouseInterface)

Select Case nReturnCode

Case MI_NODOSMOUSEDRIVER
      cMouseInterface = "Unknown"

Case SISDK_SUCCESS

      Select Case nMouseInterface

          Case MI_BUS
              cMouseInterface = "Bus"

          Case MI_SERIAL
              cMouseInterface = "Serial Port"
```

```
                Case MI_INPORT
                    cMouseInterface = "Inport"

                Case MI_PS2
                    cMouseInterface = "PS/2"

                Case MI_HP
                    cMouseInterface = "Hewlett Packard"

            End Select

      End Select

      Print "Mouse interface: " + cMouseInterface
```

Output

```
Mouse interface: InPort
```

Return Codes

CODE	VALUE	MEANING
MI_NODOSMOUSEDRIVER	-2	The mouse interface is unknown or DOS mouse driver has not been loaded prior to starting Windows.
SISDK_SUCCESS	0	The function completed successfully.

SIGetWindowsMouseDriverType

Description This function determines the Windows mouse driver type.

Syntax **INTEGER SIGetWindowsMouseDriverType()**

Returns An integer indicating the Windows mouse driver type.

Example

```
'The following code demonstrates determining the Windows
'mouse driver. For this example, the DOS mouse driver is
'Microsoft.

Dim cMouseDriver As String
Dim nMouseDriver As Integer

nMouseDriver = SIGetWindowsMouseDriverType()

Select Case nMouseDriver

    Case MD_NOWINDOWSMOUSEDRIVER
        cMouseDriver = "No Windows Mouse Driver"

    Case MD_MICROSOFT
        cMouseDriver = "Microsoft"

    Case MD_HP
        cMouseDriver = "Hewlett Packard"

    Case MD_LOGITECH
        cMouseDriver = "Logitech"

    Case MD_OLIVETTIORATANDT
        cMouseDriver = "Olivetti / AT&T"

    Case MD_MOUSESYSTEMS
        cMouseDriver = "Mouse Systems"

End Select

Print "Windows mouse driver type: " + cMouseDriver
```

Output

```
Windows mouse driver type: Microsoft
```

Return Codes

CODE	VALUE	MEANING
`MD_NOWINDOWSMOUSEDRIVER`	-3	There is no Windows mouse driver loaded.
`MD_MICROSOFT`	1	The Windows mouse driver type is Microsoft.
`MD_HP`	2	The Windows mouse driver type is HP.
`MD_LOGITECH`	3	The Windows mouse driver type is Logitech.
`MD_OLIVETTIORATANDT`	4	The Windows mouse driver type is Olivetti or AT&T.
`MD_MOUSESYSTEMS`	5	The Windows mouse driver type is Mouse Systems.

SIGetMouseIRQ

Description This function retrieves the IRQ (interrupt) used by the mouse.

Syntax **INTEGER SIGetMouseIRQ**(*nMouseIRQ* **As Integer**)

Parameter	**Description**
nMouseIRQ	Upon successful completion, *nMouseIRQ* will contain a value from 0 to 15.

Returns An integer indicating the success or failure of the function call.

Remark The successful completion of this function requires a DOS mouse driver be loaded prior to starting Windows.

NOTE: There have been unconfirmed reports that loading a DOS mouse driver high will cause this function to behave erratically. This will be fixed in the next release.

Example

```
'The following code demonstrates determining the mouse IRQ
'For this example, the mouse IRQ is 2.

Dim cMouseIRQ As String
Dim nMouseIRQ As Integer
Dim nReturnCode As Integer

nReturnCode = SIGetMouseIRQ(nMouseIRQ)

Select Case nReturnCode

    Case MD_NODOSMOUSEDRIVER
        cMouseIRQ = "Unknown"

    Case SISDK_SUCCESS
        cMouseIRQ = Trim$(Str$(nMouseIRQ))

End Select

Print "Mouse interrupt: " + cMouseIRQ
```

Output

```
Mouse interrupt: 2
```

Return Codes

CODE	**VALUE**	**MEANING**
MI_NODOSMOUSEDRIVER	-2	There is no DOS mouse driver loaded, therefore the mouse IRQ cannot be determined.
SISDK_SUCCESS	0	The function completed successfully.

SIGetDOSMouseDriverVersion

Description This function retrieves the DOS mouse driver version.

Syntax **INTEGER SIGetDOSMouseDriverVersion**(*sDOSMouseDriverVersion* **As Single**)

Parameter	Description
sDOSMouseDriverVersion	Upon successful completion, *sDOSMouseDriverVersion* will contain the DOS mouse driver version.

Returns An integer indicating the success or failure of the function call.

Remark The successful completion of this function requires a DOS mouse driver be loaded prior to starting Windows.

NOTE: There have been unconfirmed reports that loading a DOS mouse driver high will cause this function to behave erratically. This will be fixed in the next release.

Example

```
'The following code demonstrates determining the DOS mouse
'driver version.  For this example, the mouse driver version is
'8.20

Dim sDOSMouseDriverVersion As Single
Dim nReturnCode As Integer

nReturnCode =SIGetDOSMouseDriverVersion(sDOSMouseDriverVersion)

Select Case nReturnCode

    Case MD_NODOSMOUSEDRIVER
        cDOSMouseDriverVersion = "No DOS Mouse Driver"

    Case SISDK_SUCCESS
        cDOSMouseDriverVersion =
      Trim$(Format$(sDOSMouseDriverVersion,"0.00"))

End Select

Print "DOS mouse driver version: " + cDOSMouseDriverVersion
```

Output

```
DOS mouse driver version: 8.20
```

Return Codes

CODE	VALUE	MEANING
`MI_NODOSMOUSEDRIVER`	-2	There is no DOS mouse driver loaded, therefore the mouse driver version cannot be determined.
`SISDK_SUCCESS`	0	The function completed successfully.

SIGetDOSMouseDriverType

Description This function determines the DOS mouse driver type.

Syntax **INTEGER SIGetDOSMouseDriverType(***nDOSMouseDriverType* **As Integer)**

Parameter	**Description**
nDOSMouseDriverType	Upon successful completion, *nDOSMouseDriverType* will contain one of the following driver types.

CODE	VALUE	MEANING
`MD_MICROSOFT`	1	Microsoft
`MD_HP`	2	Hewlett Packard
`MD_LOGITECH`	3	Logitech

Returns An integer indicating the success or failure of the function call.

Remark This function will only recognize Microsoft, HP, & Logitech mouse drivers and will return any other types as `SISDK_FAIL` .

NOTE: There have been unconfirmed reports that loading a DOS mouse driver high will cause this function to behave erratically. This will be fixed in the next release.

Example

```
'The following code demonstrates determining the DOS mouse
'driver. For this example, the DOS mouse driver is Microsoft

Dim cMouseDriver As String
Dim nMouseDriver As Integer
Dim nReturnCode As Integer

nReturnCode = SIGetDOSMouseDriverType(nMouseDriver)

Select Case nReturnCode

    Case MD_NODOSMOUSEDRIVER
        cMouseDriver = "No DOS Mouse Driver"

    Case SISDK_FAIL
        cMouseDriver = "Unknown"

    Case SISDK_SUCCESS

      Select Case nMouseDriver

          Case MD_MICROSOFT
              cMouseDriver = "Microsoft"

          Case MD_HP
              cMouseDriver = "Hewlett Packard"
```

```
                Case MD_LOGITECH
                    cMouseDriver = "Logitech"

            End Select

      End Select

      Print "DOS mouse driver type: " + cMouseDriver
```

Output

```
DOS mouse driver type: Microsoft
```

Return Codes

CODE	VALUE	MEANING
`MI_NODOSMOUSEDRIVER`	`-2`	There is no DOS mouse driver loaded, therefore the mouse driver version cannot be determined.
`SISDK_FAIL`	`-1`	There is a DOS mouse driver loaded, but the could not identify it.
`SISDK_SUCCESS`	`0`	The function completed successfully.

SIGetWindowsVersion

Description This function determines the version of Windows running.

Syntax **SINGLE SIGetWindowsVersion**()

Returns An single precision floating point number indicating the version of Windows.

Example

```
'The following code demonstrates determining the version of
Windows.  For this example, the version of Windows is 3.10.

Print "Windows version: " +
      Trim$(Format$(SIGetWindowsVersion(), "0.00"))
```

Output

```
Windows version: 3.10
```

SIGetWindowsFreeMemoryKB

Description This function determines the amount of free memory of Windows.

Syntax **LONG SIGetWindowsFreeMemoryKB**()

Returns An long (32-bit) integer indicating the amount of free memory in Windows.

Example

```
'The following code demonstrates determining the free memory
'of Windows.  For this example, the free memory is 6734 KB.

Print "Windows free memory: " +
      Trim$(Str$(SIGetWindowsFreeMemoryKB())) + " KB"
```

Output

```
Windows free memory: 6743 KB
```

SIGetWindowsMode

Description This function determines the mode that Windows is operating in.

Syntax **INTEGER SIGetWindowsMode**()

Returns An integer indicating the mode that Windows is operating in.

Example

```
'The following code demonstrates determining the mode of
'Windows.  For this example, Windows is running in 386 Enhanced
'Mode.

Dim nWindowsMode As Integer

Dim cWindowsMode As String

nWindowsMode = SIGetWindowsMode()

Select Case nWindowsMode

    Case MO_STANDARD
        cWindowsMode = "Standard Mode"

    Case MO_ENHANCED
  cWindowsMode =  "386 Enhanced Mode"

End Select

Print "Windows mode: " + cWindowsMode
```

Output

```
Windows mode: 386 Enhanced Mode
```

Return Codes

CODE	VALUE	MEANING
MO_STANDARD	1	Windows is running in standard mode.
MO_ENHANCED	2	Windows is running in enhanced mode.