Questions and Answers

Q: Can you provide more info on how the READ/WRITE device control string should work. Does the read device bytes command get information that was written by the write device control string?

A: At present, there are very few people that use this command. There are a couple other features that are either used only in special instances or not at all at present. This is not to say that they won't be used at some later time.

The purpose of these commands was to allow a standard way of delivering commands that were not specified in the CD-ROM device driver spec to the drive. For example, sending SCSI command strings and reading the responses from the drive. This function is deliberately open-ended and vague because it was intended to provide a catch-all mechanism for application programs to communicate requests or request data in ways that were not specified by the device driver spec. For application programs to use these functions they have to know the driver supports these functions and also how to communicate with that specific drive. The mechanism would let the driver do what it does best and worry about which ports and interrupts to use. This relieves the application program from these details and allow it to deal with controlling the device at a higher level.

Right now, if the driver does not support these functions, it should return an error for Unknown Command. One could test whether these two function were supported this way. Note: if there are commands which you feel should be supported by the device driver specification, please communicate them to us and we will consider adding them if they are of sufficient general interest.

Q: How can an application access CD-ROM drives that are subunits of one driver? The IOCTL calls do not take an argument for subunit. MSCDEX seems to handle this OK since when I do a directory of each CD-ROM in turn it accesses the correct drive. I do not see any clean way for an application to, for example lock the door on CD-ROM drive G: which is the third drive handled by the driver.

A: Requests all have a sub-unit field in the request header. Commands that one would expect to be directed to a specific drive, such as open door, are targeted at a particular drive through the use of the sub-unit field.

Q: Why doesn't MSCDEX allow IOCTL access via the drive letter (that is, MS-

DOS func. 44h subfunc. #4,5), as if the CD-ROM were a normal drive. I understand that the driver is not a block device, but this is being handled already in some way since you allow a user to perform file I/O to a CD-ROM making it appear to be a block device. It would seem that all that would be necessary to accomplish this is to intercept IOCTL calls in the same way that file access calls are being intercepted.

A: MSCDEX doesn't presently hook INT 21h, which is what this would involve. It's doubtful that this will change. It's not that much more difficult to open the file and send an IOCTL to the handle. File access calls are not caught at an INT 21h level but are caught from within MS-DOS at another interface. CD-ROM drives are far more like network drives than traditional MS-DOS FAT file structure block drives and their drivers. For example, try to FORMAT a CD-ROM drive and you'll see. Part of all this prevents IOCTL's to the drive letter from being directed to the appropriate driver. For more information, see the section “MS-DOSifying Your CD-ROM.”

Q: Why not allow access to the PLAY, STOP and SEEK functions via the INT 2Fh entry point as is allowed for READ LONG. This would be much simpler than requiring the application to locate the driver header and then find the STRATEGY entry point and create request control blocks etc. This is a lot of code to start the music playing!

A: The reason we haven't included play and other commands of this type in the INT 2Fh interface is to avoid loading down MSCDEX with additional functionality that most people don't use. Your suggestion would only move that code from the CD-playing program into MSCDEX. It makes your program smaller at the expense of making MSCDEX larger.

Q: Shouldn't the specification eliminate the need for the application to OPEN the driver by name? This is especially important in systems where the driver creates a new driver header for each CD-ROM drive. As it is, MS-DOS allows so few file handles to be open simultaneously that requiring applications to open even more is very bad.

A: Simply close the driver handle after you have located the device header. You no longer need to communicate through MS-DOS to control it, so free the handle and make it available for other programs to use. With version 2.10, it is no longer necessary to OPEN the device driver in order to communicate with it. Applications can communicate with the device driver using *Send Device Driver Request.*

Q: It seems that there should be bits in the Device Status word to indicate whether a driver supports Reading/Writing device control strings.

A: Reading and writing device control strings was put in as a catch-all for anything that was missed so that application programs could send specific commands through the device driver to the device if they understood the device and knew how to communicate to it. A manufacturers CD-ROM diagnostic program would be an example of a program that might choose to communicate with the drive in this way. If the driver does not support these functions, it should return an error. One can test whether these two function are supported by testing if the error returned is for *Unknown Command.*

Q: In the spec, treatment of the BUSY bit in the status word with regard to the PLAY AUDIO function seems to assume only one CD-ROM drive. What happens when the user has two or more drives each of which want to be playing music? How does the application tell whether the desired drive is busy? It would seem better to use some of the bits in the upper word of Device Status to indicate BUSY for each drive, perhaps allowing 8 or 16 drives.

A: Requests all have sub-unit numbers associated with them. A request for service from one sub-unit may report that the drive is busy at the same time another sub-unit was not busy. The sub-unit field is used to distinguish requests between the drives supported by the driver. The busy bit serves as an indication of drive status for the sub-unit the request is for.

Q: In the device driver spec, it says that if more than one unit is supported by the driver that this field should be set to the number of units. I suspect that this is wrong since this is not a block device. As far as I can see, this field should only ever be set to one since each unit will actually have its own header with its own unique name.

A: CD-ROM device drivers are a hybrid of block and char device drivers and are not technically legal as one or the other. Block drivers make some assumptions about the media format that aren't meaningful for CD-ROM and don't have a read call that can deal with CD-ROM's large data space. They were made as char devices with some additional calls and rules. One of the changes that was made for CD-ROM device drivers was to allow multiple sub-units for the device so the treatment of this field is correct as specified even though CD-ROM device drivers are character device drivers.

If one has more than one CD-ROM drive, one can approach supporting them from several ways. One could have separate device drivers for each drive and load one per drive, have a single driver with multiple device headers, or have a single driver with one device header that supports sub-units. This last method is borrowed from block drivers. For the case that the drives and drive commands are all the same, using sub-units will allow you to distinguish between which drive receives which command. The alternatives clutter MS-

DOS up with drivers or device headers. Sub-units may not be legal character device driver fields but conceptually, they're the right thing. Since CD-ROM device drivers could not be block drivers and had to be char device drivers, some liberties were taken with the specification to merge the best of both specifications.

Q: Is there any support through MSCDEX for WRITE LONG? I have a need for this to support a CD mastering system. I would like to be able to treat a WORM drive as a CD-ROM and allow writing to the drive once to create a master and then be able to test it out by using it as CD-ROM to verify that our data has been correctly stored in High Sierra format.

A: Such a call exists. It only serves to define a standard interface for CD-ROM device drivers that are running over re-writable media - such as a CD mastering system. It is in the driver specificiation starting with version 2.00 of the CD-ROM Extensions.

Q: How important is it that I should support RAW mode access in my driver? What would this typically be used for?

A: This is something that is gaining in importance. Several drives support reading in raw mode. Since drives and their command capabilities are hardware dependent, you would know based on the capabilities of your hardware if you were capable of supporting it. This function was added for completeness. A standard way was needed to define how to get at the 288 bytes of EDC/ECC if drives allowed access and to have an avenue prepared if people found useful applications that would not use EDC/ECC where they wanted the additional space (such as gracefully degrading low-fidelity audio or graphics). It will be useful for copying audio information or for audio systems that will want to be able to manipulate audio tracks. Many people have expressed interest in having this capability.

Q: In the structure for the UPC Code function, "The UPC/EAN code (last 4 bits are zero)". Does this mean the low order or high order 4 bits?

A: This is less ambiguous if you read Redbook under mode-2 of the Q-channel info. This is now clarified in the UPC Code call. It should be the low nibble of byte 7. Redbook specifies that MSB comes out first so the high nibble will contain the 13th nibble of the UPC code and the 14th nibble will be zero.

Unfortunately, scanning for the UPC code is time consuming especially if it is done by the software. This is due to the design of the q-channel in Redbook. It's a pity because this could be a useful number to verify the correct disc has been inserted. Most CD-ROMs do not have a UPC code or have it zeroed out.

The same seems to be true for CD-audio as well. We believe that CD-ROM drives should scan for the UPC code as they read the Table of Contents when initializing from power up or a new disc. If the hardware does not do this, the UPC code has to be scanned by polling the q-channel which may occasionally miss the UPC code.

Q: It would be nice if the device driver spec included a list of what types of disk access functions would and would not work so that users could get an idea of what utilities and applications will and will not work with the extensions.

A: The device driver specification describes just what is necessary for writing a CD-ROM device driver. The information you would like concerning things such as INT 25h/26h not supported as well as the behavior CHKDSK/FORMAT/etc belongs and is mentioned in the MSCDEX overview.

Q: If I have a low priority request and if the system has time, can the prefetch request read into and transfer the data into a transfer address?

A: We have looked at this for some time but the bottom line is that asynchronous I/O under MS-DOS is very difficult to support in all cases. It is difficult for MSCDEX or the CD-ROM device driver to know that the transfer address is still valid because MS-DOS never notifies MSCDEX or the device driver if the requesting process was been terminated. The request runs the risk of writing over another program. The best approach now is if the driver wants to, it can reserve internal buffer space for data from the disc and put prefetched data there. Then it can copy the data to the read transfer address once the read request finally arrives. Alternately, some of the caching or prefetching can reside in the CD-ROM controller or in the drive itself.

Q: Is there any status indication that a prefetch transfer has occurred or some interaction with the READ LONG command?

A: There is no way to tell if a prefetch request was successful or the state of it. The prefetch simply provides a hint to the driver and the read request later is the request that finally takes delivery of the data.

Q: My driver seems to work ok except that whenever I connect to a subdirectory and do a directory, I am suddenly back in the root directory again. What's going wrong?

A: What is most likely happening is the driver is returning an incorrect value for MEDIA CHECK and MSCDEX thinks that the disc is changing all the time. When this happens, MSCDEX rereads the volume descriptors and pathtable and reinitializes what it knows about the disc and changes the current working directory back to root as if the drawer had been opened, the disc removed, and then reinserted. This will be accompanied with a larger amount of disc activity than one would expect for a simple directory scan. Fixing the driver to return the correct value when asked for a media check will correct this behavior.

Q: What is the best way for my application to know if the disc has changed since it was last accessed?

A: Use the MS-DOS function find first and look for the volume id. When the disc has been read and MSCDEX has already initialized the internal information it keeps for each disc, this is a relatively inexpensive operation. The information is in memory and the disc does not have to be touched, so checking the volume id is very quick. Only if the disc has been changed does the disc have to be touched. This operation takes considerably longer than if the disc was not changed but even so, this has to be done anyway because MSCDEX has to read and initialize what it knows about the new disc so it can report the volume id correctly so the application can know if the disc in the drive is the one that it is looking for.

Q: When I do a directory, the first couple filenames are either duplicated or they are random characters. What might cause this?

A: The problem comes from having the incorrect bytes in the file identifier field for the first two directory entries. The first directory entry in each directory file is supposed begin with a copy of the directory record for that directory file followed by a copy of the directory record for the parent directory (also known as '.' and '..' on Unix or MS-DOS). The filename or directory identifier is supposed to be 1 byte long and the contents are supposed to be 0 for the first directory entry and 1 for the second directory entry. This is discussed in clause 6.8.2.2 of the ECMA standard or the ISO-9660 proposal. Several places on the disc in question, you have a 1 where there should be a 0 and in one directory, the file identifier consists of 0x8A which is why DIR in that directory begins with an "e". Incorrectly formatted discs will not be handled by the extensions correctly. This is why it is a good idea to test your disc image using MSCDEX before you press a disc to make sure your data is formatted correctly and as MSCDEX expects it.

Q: I have a directory file that is 4 kilobytes long but when I do a DIR in that directory, it is slower than usual and random filenames are printed out. I can tell by watching the device driver commands that MSCDEX is asking for sectors far beyond the end of the directory. I can see how this might account for the random filenames but why is it scanning so far?

A: Problems such as this result from having with multi-sector directory files that include empty sectors in the directory file. The High Sierra specification does not allow you to have empty directory sectors at the end or to have gaps in the middle. However, ISO9660 does not have this constraint. (This will be addressed in a later version.) The problem stems from the fact that your directory length is too long. For example, for the disc in question, the root directory begins at sector 28 and its length is 4096 bytes but the second sector is completely blank (all 0's). This confuses MSCDEX because it does not expect to see empty sectors.

Q: I noticed that Function Request 0 Get Number of CD-ROM drive letters may not always return unambiguous results. Suppose I start the network first and use one of the drive letters for a network drive (F:). When I start the Extensions, it will begin assigning drive letters after the last used drive letter (C: on my machine). If I have 4 CD-ROM drives on my system, they will be assigned drive letters D:, E:, G:, and H:. Function 0 returns 4 in BX for the number of CD-ROM drives and 3 in CX for drive letter D: correctly. But as you can see, the CD-ROM drives do not use contiguous drive letters so I cannot deduce from what this function returns that drive F: is not a CD-ROM drive.

A: That is correct. This is why function 0Dh Get CD-ROM drive letters was added. To get an unambiguous list of CD-ROM drives, use this function or use function 0Bh CD-ROM Drive Check to tell if a drive letter is for a CD-ROM drive.

Q: Is it possible to do an absolute read using the Extensions. I am trying to read mode 2 (uncooked) data using Function Request 8 Absolute Read. I use a normal device I/O to turn off error correction and perform a read but all I get back is 2048 bytes of data instead of the full 2356 bytes. Is there another way in INT 2Fh to get the data uncooked?

A: Not at present. If you want to get at the data including error correction code, you will have to communicate directly with the device driver. The Extensions provides the *Send Device Driver Request* mechanism for communicating with the device driver.

Q: Is it possible to access a non-High Sierra disc with the Extensions using an absolute disc read?

A: One can use either the extensions to read a non-High Sierra disc using INT 2Fh or one can communicate directly with the device driver to do this. The device driver itself makes no distinction between High Sierra and non-High Sierra discs so it can be used to read them although the burden of file system translation and reading then falls on the application talking to the driver. The

INT 2Fh Absolute Read function simply packages the request to read and sends it directly to the driver and returns the result.

Q: What we have done is, in AUTOEXEC.BAT, first loaded the MS-DOS CD-ROM Extensions and then the MS-NET software. The error message is "Redirector already installed". The network software is then not loaded. We are using MS-NET 1.1 in an HP product called ThinLAN. Any hints as to what they should try next?

A: MSCDEX is a CD-ROM "redirector". It hooks into MS-DOS the same way the network redirector does to get requests for file access to files that are not on local hard/floppy disks. As far as MS-DOS is concerned, CD-ROM drives look just like network drives. MS-DOS passes all file accesses through the redirector interface to the network redirector which in turn sends file access requests out over the net. MSCDEX splices itself in front of the network redirector and takes requests belonging to CD-ROM drives and passes the rest to the network redirector.

The problem is that the network redirector code assumes that there will only be one redirector installed (itself) whereas MSCDEX does not make this assumption. If the network redirector is installed after MSCDEX (before it in the interrupt chain), it will process all requests from MS-DOS and never pass any CD-ROM requests through to MSCDEX. For this reason, MSCDEX has to be installed after the network redirector (before it in the interrupt chain) and so MSCDEX prevents the network redirector from installing afterwards to ensure this. Since you installed MSCDEX first, the network believes a redirector is already installed so it does not install itself which is what you are seeing. In order to install both, simply install your network software first and MSCDEX second and you're set.

Q: CHKDSK, ASSIGN, and SUBST report that the CD-ROM is a network disc. Why is this?

A: From the above explanation, you understand that to MS-DOS, the CD-ROM drives look like network drives. The programs CHKDSK, ASSIGN, and SUBST check the same fields MS-DOS does and think the same thing. There is no way to get around this.

Q: RENAME gives error message "Duplicate file name or File not found" instead of something that makes sense such as "Access denied" or "Can't rename CD-ROM files".

A: The error message is coming from the code for RENAME and not MSCDEX. The error condition is being returned correctly but the error code returned by version 1.01 is correct according to MS-DOS documentation. The problem seems to be that there are two error codes for access denied - 5 and a special one 65 which is error_net_access_denied which is returned by the network redirector when it has a problem. MSCDEX version 2.00 and up returns error code error_net_access_denied and so RENAME now reports "Access denied".

Q: Why does the SETVER command have to be used when MSCDEX 2.20 is used with MS-DOS 5.0?

A: MSCDEX 2.20 was completed in the early phases of MS-DOS 5.0 development, thus, the capabilities of MS-DOS 5.0 were not available for MSCDEX development. While MSCDEX 2.20 was developed in advance of MS-DOS 5.0, its release however,was delayed until a few months before that of MS-DOS 5.0.

Q: Why can't I load MSCDEX 2.21 in high memory using MS-DOS 5.0?

A: MSCDEX Version 2.21 cannot load high because it is essentially an upgrade to improve the interface between it and MS-DOS 5.0. A future release of MSCDEX will take advantage of the features available in MS-DOS 5.0.

While you cannot load MSCDEX 2.21 in high memory, you can still use the /E command line switch to have MSCDEX use the expanded memory installed in your system. This moves a portion of MSCDEX into expanded memory which frees some of the lower memory normally used by MSCDEX.

Q: Where can I get more information on CD-AUDIO, CD-ROM devices, and MS-DOS device drivers?

A: ISO 9660 specifies the standards of the volume and file structure used by CD-ROM discs. ISO 10149 specifies the data characteristics (such as the format of the tracks, error-detecting and error-correcting characters, and coding of information) of CD-ROM discs. IEC 908 specifies the standards used by CD digital audio system. These specifications are available from ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission).

For information about MS-DOS drivers, you will find the following books helpful:

Duncan, Ray. *Advanced MS-DOS Programming*. Redmond: Microsoft Press, 1988

Microsoft MS-DOS Programmer's Reference, Redmond: Microsoft Press, 1991.

Also, the MS KnowledgeBase topics of The Microsoft Programmer's Library CD-ROM contain answers to many of the common questions asked about programming under MS-DOS.

Q: What is the current support for CDROM XA in MSCDEX?

A: CDROM XA is a novel definition of the use of Mode 2 sectors on a CD. MSCDEX supports plain ISO 9660 file access to files with XA system use fields. It is the responsibility of the driver and hardware to process or control the XA subsystems. A driver that has such specialized support will have bit 10 set in the DEVICE_STATUS (IOCTL Read (03h) subfunction 06h) return value.

End.