Some notes on Video Digitizers since QuickTime 1.0

The big effort here has been to add support to handle cards with capabilities previously unaddressed. Some hints calls have been added as well.

A new output capability flag has been added: digiOutDoesUnreadableScreenBits. If the vdig sets this bit, it means that the vdig can put pixels on the screen for the user to see, but that these pixels cannot be used for compressing images. This occurs with several cards. It this bit is set, it is also assumed that the card can continue to do play through while digitizing images to another destination. If anyone can prove this to be a bad assumption, we can change it. Speak up soon.

A bunch of stuff is being added to deal with cards that can provide compressed video images directly.

The flag digiOutDoesCompress indicates that the card can provide some kind of compressed data. The flag digiOutDoesCompressOnly indicates that it can only provide compressed data, that no straight RGB pixels are ever available. The flag digiOutDoesPlayThruDuringCompress indicates that the card can be drawing pixels to the screen at the same time it is providing compressed data.

A bunch of new calls for vdigs have been defined to deal with compressed sources. They are described below.

```
ComponentResult VDSetCompression,(VDIG , OSType compressType, short depth, Rect *bounds,
            CodecQ spatialQuality, CodecQ temporalQuality, long keyFrameRate)
```

This routine tells the vdig what sort of compression is requested. The parameters are identical to those found throughout the Image Compression document (and SGSetVideoCompression call) so I won't get into it here. If the compression requested can't be supported, return an error.

If this call is made with compressType, depth, and bounds set to 0, the settings for spatial and temporal qualities and key frame should be applied to the currently active compression stream. This allows a client to vary the data rate without having to stop grabbing and reset everything.

```
ComponentResult VDCompressOneFrameAsync(VDIG )
```

This call starts the grab of a compressed frame.

```
ComponentResult VDCompressDone(VDIG , Boolean *done, Ptr *theData, long *dataSize, unsigne
            char *similarity, TimeRecord *t))
```

This call should return true in "done" if the compressed frame is done. If it returns true, it must fill in the remaining parameters as well. Currently, the VDIG is responsible for allocating the memory for the compressed data to go into.

The pointer returned must be accessible by the client in the current memory mode. This means that if the machine is running in 24–bit mode, you can’t return a buffer in 32–bit NuBus card memory. However, when running in 32–bit mode, this is not a problem.

The TimeRecord should be filled in when the compress is done. It should be a complete TimeRecord indiciating the time at which the returned frame was grabbed. It should be based on the TimeBase provided in the VDSetTimeBase call. It is important not to return frames out of time order. If the TimeBase has a non-zero rate it is also important not to return two frames with the same time.

```
pascal VideoDigitizerError VDReleaseCompressBuffer (VDIG , Ptr bufferAddr)
```

When the client is done with the data returned from VDCompressDone, the must call VDReleaseCompressBuffer to inform the vdig that they are done copying out the contents of the buffer. At this point, vdig may start putting more data into the buffer. The address passed VDReleaseCompressBuffer should only be those returned by VDCompressDone.

By having the vdig allocate the storage to compress into, the vdig is completely in control of having multiple buffers, and how those buffers are managed.

```
ComponentResult VDGetImageDescription(VDIG , ImageDescriptionHandle desc)
```

Fills out the provided handle with an ImageDescription for the current compression settings.

```
ComponentResult VDResetCompressSequence(VDIG )
```

If frame differenced compression is taking place, it may sometimes be necessary to force a key frame. When VDResetCompressSequence is called, the VDIG should ensure that the next returned frame is a key frame.

```
pascal VideoDigitizerError VDSetCompressionOnOff (VDIG , Boolean state)
```

This call is used to turn on and off compression. If a vdig only can supply compressed data, this call is unnecessary but may be called anyway, so it should be implemented. This call can be used to turn compression off, when it is no longer required. This call will precede any calls to VDSetCompression or VDCompressOneFrameAsync.

```
pascal VideoDigitizerError VDGetCompressionTypes (VDIG , VDCompressionListHandle h)
```

This call returns a handle filled with information about the compression capabilities of the digitizer. The handle passed in must be resized to accommodate the necessary information. The handle should be filled in with an array of VDCompressionList structures. If the vdig supports multiple types of compression, its preferred (default) compression should be first in the list. While it is not necessary to have a Compressor Codec installed to capture a given type of compressed data from a vdig, it is essential that a Decompressor Codec for the given type be available in the system.

```
typedef struct VDCompressionList {
      CodecComponent          codec;
      CodecType               cType;
      Str63                   typeName;
      Str63                   name;
      long                    formatFlags;
      long                    compressFlags;
      long                    reserved;
} VDCompressionList, *VDCompressionListPtr, **VDCompressionListHandle;
```

The codec field will typically be nil. However, if you provide a compressor codec that can be communicated with as well, you should return it here. The cType field is the type of data you will be providing. The typeName, name, formatFlags and compressFlags should correspond to the GetCodecInfo data structures. The reserved field should be set to zero.

```
pascal VideoDigitizerError VDSetTimeBase (VDIG , TimeBase t)
```

The given TimeBase should be used to provide any future times. In particular, the TimeBase should be used to return frame times in VDCompressionDone.

```
pascal VideoDigitizerError VDSetFrameRate (VDIG , Fixed framesPerSecond)
```

This call provides a hint to the Video Digitizer as to the desired frame rate of the application. If the Video Digitizer can’t provide any special frame limiting services, or you

don't implement this call, return "digiUnimpErr" and the Sequence Grabber will emulate this feature in software. If framesPerSecond is set to zero, it means you should operate at your default frame rate - typically 30 frames per second, if possible.

```
pascal VideoDigitizerError VDGetDataRate (VDIG, long *milliSecPerFrame, Fixed
            *framesPerSecond, long *bytesPerSecond)
```

This call provides hints to the caller about some of the performance characteristics of the Video Digitizer. You can return any of the parameters that you have values for. Any unknown parameters should be set to zero. The values returned should be based on the current digitizer settings, and the machine currently running on. These values are used as hints for allocating buffers and bandwidth.

The "milliSecPerFrame" should contain the number of milliseconds overhead involved in capturing each frame. This should include the average delay between requesting a frame, and its becoming available; and any overhead incurred in reading the bits from the frame buffer.

The "framesPerSecond" parameter should return the approximate frame rate that the Video Digitizer will be able to grab frames at. This number may or may not be the same as that provided by SGSetFrameRate.

The "bytesPerSecond" parameter is only used for Video Digitizers that return compressed data. This allows you to return the average data rate for grabbing given the current settings.

```
pascal VideoDigitizerError VDGetSoundInputDriver (VDIG , Str255 soundDriverName)
```

Yet another hints call in the Video Digitizer interface. If the Video Digitizer hardware has an associated Sound Input Device, it should return the name of that Sound Input Driver from this call. In this way, applications can provide the user with a good default sound device when opening up a new Sequence Grabber.

If your Video Digitizer is not associated with any particular Sound Input Device you can return "digiUnimpErr" for this call.