

This chapter discusses sequence grabber components. **Sequence grabber components** allow applications to obtain digitized data from external sources. Applications can then request that the sequence grabber display that data or store it in QuickTime movie files. If you are writing an application that needs to acquire data from sources external to the Macintosh computer, or if you are developing a sequence grabber channel component, you should read this chapter. If you are developing a channel component, you should also read the chapter “Sequence Grabber Channel Components.”

Note that the information in this chapter is presented from the perspective of a developer of an application that uses sequence grabber components. If you are developing a sequence grabber component, your component must support the interfaces described in this chapter.

This chapter has been divided into the following sections:

- “About Sequence Grabber Components” presents general information about sequence grabber components.
- “Using Sequence Grabber Components” discusses how to use the sequence grabber component to preview and record captured data. It then provides a sample program that shows how to play captured data and save it in a QuickTime movie.
- “Sequence Grabber Components Reference” describes the constants and data structures that an application needs to communicate with sequence grabber components as well as the functions that your sequence grabber component must support.
- “Summary of Sequence Grabber Components” supplies a summary of the sequence grabber component constants, data types, and functions in C and in Pascal.

## About Sequence Grabber Components

---

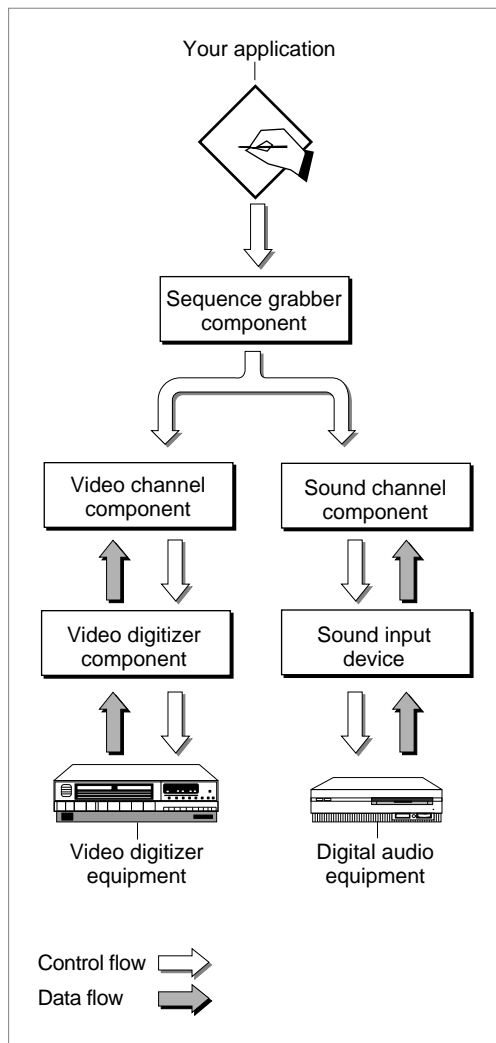
Sequence grabber components allow applications to obtain digitized data from sources that are external to a Macintosh computer. For example, you can use a sequence grabber component to record video data from a video digitizer. Your application can then request that the sequence grabber store the captured video data in a QuickTime movie. In this manner, you can acquire movie data from various sources that can augment the movie data you create by other means, such as computer animation. You can also use sequence grabber components to obtain and display data from external sources, without saving the captured data in a movie.

The sequence grabber component provided by Apple allows applications to capture both audio and video data easily, without concern for the details of how the data is acquired. When capturing video data, this sequence grabber uses a video digitizer component to supply the digitized video images (see the chapter “Video Digitizer Components” in this book for more information about video digitizer components). When working with audio data, Apple’s sequence grabber component retrieves its sound data from a sound input device (see *Inside Macintosh: More Macintosh Toolbox* for more information about sound input devices).

## Sequence Grabber Components

Sequence grabber components use sequence grabber channel components (or, simply, channel components) to obtain data from the audio- or video-digitizing equipment. These components isolate the sequence grabber from the details of working with the various types of data that can be collected. The features that a sequence grabber component supplies are dependent on the services provided by sequence grabber channel components. The channel components, in turn, may use other components to interact with the digitizing equipment. For example, the video channel component supplied by Apple uses a video digitizer component. Figure 5-1 shows the relationship between these components and your application.

**Figure 5-1** Relationships among your application, a sequence grabber component, and channel components



Sequence grabber panel components augment the capabilities of sequence grabber components and sequence grabber channel components by allowing sequence grabbers to obtain configuration information from the user for a particular digitizing source. Sequence grabbers present a settings dialog box to the user whenever an application calls the `SGSettingsDialog` function (see “Working With Sequence Grabber Settings” beginning on page 5-47 for more information about this sequence grabber function). Applications never call sequence grabber panel components directly; application developers use panel components only by calling the sequence grabber component. See the chapter “Sequence Grabber Panel Components” in this book for more information about the sequence grabber configuration dialog box and the relationships of sequence grabbers, sequence grabber channels, and sequence grabber panels.

If you are developing digitizing equipment and you want to allow applications to use the services of your equipment with a sequence grabber component, you should create an appropriate video digitizer component or sound input device driver. See the chapter “Video Digitizer Components” later in this book for a description of video digitizer components. See *Inside Macintosh: More Macintosh Toolbox* for information about sound input device drivers.

If you are developing equipment that provides a new type of data to QuickTime, you should develop a new sequence grabber channel component. See the chapter “Sequence Grabber Channel Components” in this book for a complete description of sequence grabber channel components.

## Using Sequence Grabber Components

You can use the sequence grabber component to play captured data for the user or to save captured data in a QuickTime movie. The sequence grabber component provides functions that give your application precise control over the display of the captured data.

This section describes how to use the basic sequence grabber component functions as well as the functions that allow you to configure video and sound channels.

Sequence grabber components are standard components that are managed by the Component Manager. See the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox* for more information about the Component Manager and about how to use components.

Apple has defined a component type value for sequence grabber components—that type value is `'barg'`. You can use the following constant to specify this type value.

```
#define SeqGrabComponentType 'barg' /* sequence grabber
                                   component type */
```

## Sequence Grabber Components

Apple has defined a functional interface for basic sequence grabber components. For information about the functions a sequence grabber component may support, see “Sequence Grabber Component Functions,” which begins on page 5-24.

You can use the following constants to refer to the request codes for each of the functions that a sequence grabber component may support.

```
enum {
    /* selectors for basic sequence grabber component functions */
    kSGInitializeSelect          = 0x1;    /* SGInitialize */
    kSGSetDataOutputSelect       = 0x2;    /* SGSetDataOutput */
    kSGGetDataOutputSelect       = 0x3;    /* SGGetDataOutput */
    kSGSetGWorldSelect           = 0x4;    /* SGSetGWorld */
    kSGGetGWorldSelect           = 0x5;    /* SGGetGWorld */
    kSGNewChannelSelect          = 0x6;    /* SGNewChannel */
    kSGDisposeChannelSelect      = 0x7;    /* SGDisposeChannel */
    kSGStartPreviewSelect        = 0x10;   /* SGStartPreview */
    kSGStartRecordSelect         = 0x11;   /* SGStartRecord */
    kSGIdleSelect                = 0x12;   /* SGIdle */
    kSGStopSelect                = 0x13;   /* SGStop */
    kSGPauseSelect               = 0x14;   /* SGPause */
    kSGPrepareSelect             = 0x15;   /* SGPrepare */
    kSGReleaseSelect             = 0x16;   /* SGRelease */
    kSGGetMovieSelect            = 0x17;   /* SGGetMovie */
    kSGSetMaximumRecordTimeSelect = 0x18;   /* SGSetMaximumRecordTime */
    kSGGetMaximumRecordTimeSelect = 0x19;   /* SGGetMaximumRecordTime */
    kSGGetStorageSpaceRemainingSelect = 0x1a; /* SGGetStorageSpaceRemaining */
    kSGGetTimeRemainingSelect    = 0x1b;   /* SGGetTimeRemaining */
    kSGGrabPictSelect            = 0x1c;   /* SGGrabPict */
    kSGGetLastMovieResIDSelect   = 0x1d;   /* SGGetLastMovieResID */
    kSGSetFlagsSelect            = 0x1e;   /* SGSetFlags */
    kSGGetFlagsSelect            = 0x1f;   /* SGGetFlags */
    kSGSetDataProcSelect         = 0x20;   /* SGSetDataProc */
    kSGNewChannelFromComponentSelect = 0x21; /* SGNewChannelFromComponent */
    kSGDisposeDeviceListSelect    = 0x22;   /* SGDisposeDeviceList */
    kSGAppendDeviceListToMenuSelect = 0x23; /* SGAppendDeviceListToMenu */
    kSGSetSettingsSelect         = 0x24;   /* SGSetSettings */
    kSGGetSettingsSelect         = 0x25;   /* SGGetSettings */
    kSGGetIndChannelSelect       = 0x26;   /* SGGetIndChannel */
    kSGUpdateSelect              = 0x27;   /* SGUpdate */
    kSGGetPauseSelect            = 0x28;   /* SGGetPause */
    kSGSettingsDialogSelect      = 0x29;   /* SGSettingsDialog */
    kSGGetAlignmentProcSelect     = 0x2A;   /* SGGetAlignmentProc */
    kSGSetChannelSettingsSelect   = 0x2B;   /* SGSetChannelSettings */
    kSGGetChannelSettingsSelect   = 0x2C;   /* SGGetChannelSettings */
}
```

## Sequence Grabber Components

```

/* selectors for common channel configuration functions */
kSGCSetChannelUsageSelect      = 0x80; /* SGCSetChannelUsage */
kSGCGetChannelUsageSelect      = 0x81; /* SGCGetChannelUsage */
kSGCSetChannelBoundsSelect     = 0x82; /* SGCSetChannelBounds */
kSGCGetChannelBoundsSelect     = 0x83; /* SGCGetChannelBounds */
kSGCSetChannelVolumeSelect     = 0x84; /* SGCSetChannelVolume */
kSGCGetChannelVolumeSelect     = 0x85; /* SGCGetChannelVolume */
kSGCGetChannelInfoSelect       = 0x86; /* SGCGetChannelInfo */
kSGCSetChannelPlayFlagsSelect  = 0x87; /* SGCSetChannelPlayFlags */
kSGCGetChannelPlayFlagsSelect  = 0x88; /* SGCGetChannelPlayFlags */
kSGCSetChannelMaxFramesSelect  = 0x89; /* SGCSetChannelMaxFrames */
kSGCGetChannelMaxFramesSelect  = 0x8a; /* SGCGetChannelMaxFrames */
kSGCSetChannelRefConSelect     = 0x8b; /* SGCSetChannelRefCon */
kSGCSetChannelClipSelect       = 0x8c; /* SGCSetChannelClip */
kSGCGetChannelClipSelect       = 0x8d; /* SGCGetChannelClip */
kSGCGetChannelSampleDescriptionSelect = 0x8e;
                                /* SGCGetChannelSampleDescription */
kSGCGetChannelDeviceListSelect = 0x8f; /* SGCGetChannelDeviceList */
kSGCSetChannelDeviceSelect     = 0x90; /* SGCSetChannelDevice */
kSGCSetChannelMatrixSelect     = 0x91; /* SGCSetChannelMatrix */
kSGCGetChannelMatrixSelect     = 0x92; /* SGCGetChannelMatrix */
kSGCGetChannelTimeScaleSelect  = 0x93; /* SGCGetChannelTimeScale */

/* selectors for video channel configuration functions */
kSGCGetSrcVideoBoundsSelect     = 0x100; /* SGCGetSrcVideoBounds */
kSGCSetVideoRectSelect          = 0x101; /* SGCSetVideoRect */
kSGCGetVideoRectSelect          = 0x102; /* SGCGetVideoRect */
kSGCGetVideoCompressorTypeSelect = 0x103; /* SGCGetVideoCompressorType */
kSGCSetVideoCompressorTypeSelect = 0x104; /* SGCSetVideoCompressorType */
kSGCSetVideoCompressorSelect    = 0x105; /* SGCSetVideoCompressor */
kSGCGetVideoCompressorSelect    = 0x106; /* SGCGetVideoCompressor */
kSGCGetVideoDigitizerComponentSelect
                                = 0x107;
                                /* SGCGetVideoDigitizerComponent */
kSGCSetVideoDigitizerComponentSelect
                                = 0x108;
                                /* SGCSetVideoDigitizerComponent */
kSGCVideoDigitizerChangedSelect = 0x109; /* SGCVideoDigitizerChanged */
kSGCSetVideoBottlenecksSelect   = 0x10a; /* SGCSetVideoBottlenecks */
kSGCGetVideoBottlenecksSelect   = 0x10b; /* SGCGetVideoBottlenecks */
kSGCGrabFrameSelect             = 0x10c; /* SGCGrabFrame */
kSGCGrabFrameCompleteSelect     = 0x10d; /* SGCGrabFrameComplete */

```

## Sequence Grabber Components

```

kSGCDisplayFrameSelect      = 0x10e; /* SGCDisplayFrame */
kSGCCompressFrameSelect     = 0x10f; /* SGCCompressFrame */
kSGCCompressFrameCompleteSelect = 0x110; /* SGCCompressFrameComplete */
kSGCAddFrameSelect         = 0x111; /* SGCAddFrameSelect */
kSGCTransferFrameForCompressSelect = 0x112;
                                /* SGCTransferFrameForCompress */
kSGCSetCompressBufferSelect = 0x113; /* SGCSetCompressBuffer */
kSGCGetCompressBufferSelect = 0x114; /* SGCGetCompressBuffer */
kSGCGetBufferInfoSelect     = 0x115; /* SGCGetBufferInfo */
kSGCSetUseScreenBufferSelect = 0x116; /* SGCSetUseScreenBuffer */
kSGCGetUseScreenBufferSelect = 0x117; /* SGCGetUseScreenBuffer */
kSGCGrabCompressCompleteSelect = 0x118; /* SGCGrabCompressComplete */
kSGCDisplayCompressSelect   = 0x119; /* SGCDisplayCompress */
kSGCSetFrameRateSelect      = 0x11A; /* SGCSetFrameRate */
kSGCGetFrameRateSelect      = 0x11B; /* SGCGetFrameRate */

/* selectors for sound channel configuration functions */
kSGCSetSoundInputDriverSelect = 0x100; /* SGCSetSoundInputDriver */
kSGCGetSoundInputDriverSelect = 0x101; /* SGCGetSoundInputDriver */
kSGCSoundInputDriverChangedSelect = 0x102;
                                /* SGCSoundInputDriverChanged */
kSGCSetSoundRecordChunkSizeSelect = 0x103;
                                /* SGCSetSoundRecordChunkSize */
kSGCGetSoundRecordChunkSizeSelect = 0x104;
                                /* SGCGetSoundRecordChunkSize */
kSGCSetSoundInputRateSelect   = 0x105; /* SGCSetSoundInputRate */
kSGCGetSoundInputRateSelect   = 0x106; /* SGCGetSoundInputRate */
kSGCSetSoundInputParametersSelect = 0x107;
                                /* SGCSetSoundInputParameters */
kSGCGetSoundInputParametersSelect = 0x108;
                                /* SGCGetSoundInputParameters */
/* selectors for utility functions provided to channel components */
kSGWriteMovieData            = 0x100; /* SGWriteMovieData */
kSGAddFrameReferenceSelect    = 0x101; /* SGAddFrameReference */
kSGGetNextFrameReferenceSelect = 0x102; /* SGGetNextFrameReference */
kSGGetTimeBaseSelect         = 0x103; /* SGGetTimeBase */
kSGSortDeviceListSelect      = 0x104; /* SGSortDeviceList */
kSGAddMovieDataSelect        = 0x105; /* SGAddMovieData */
kSGChangedSourceSelect       = 0x106; /* SGChangedSource */
};

```

## Previewing and Recording Captured Data

---

You can use sequence grabber components in two ways: to play digitized data for the user or to save captured data in a QuickTime movie. The process of displaying data that is to be captured is called *previewing*; saving captured data in a movie is called *recording*. You can use previewing to allow the user to prepare to make a recording. If you do so, your application can move directly from the preview operation to a record operation, without stopping the process.

### Previewing

---

Previewing captured data involves playing that data for the user as it is captured. For video data, this means displaying the video images on the computer screen. For audio data, this means playing the sound through the computer's sound system. The following paragraphs outline the steps you must follow to preview captured data.

1. First, you must open a connection to the sequence grabber component. Use the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.
2. Once you have a connection to a sequence grabber component, you must configure the component for the preview operation. Use the `SGSetGWorld` function (described on page 5-29) to set the graphics world in which the preview is to be displayed. Allocate the appropriate channels by calling the `SGNewChannel` function (described on page 5-31). You must call this function once for each channel to be used by the sequence grabber component. Use the `SGSetChannelUsage` function (described on page 5-59) to specify that each channel is to be used for previewing. You can then use the appropriate channel configuration functions to prepare the channel for the preview operation. For video channels, use the functions discussed in "Working With Video Channels" beginning on page 5-77. For sound channels, use the functions discussed in "Working With Sound Channels" beginning on page 5-92.
3. You start the preview operation by calling the `SGStartPreview` function (see page 5-37). The sequence grabber component then begins collecting data from the channels that you have created and plays that data appropriately. You can pause and restart the preview by calling the `SGPause` function (see page 5-41). Use the `SGStop` function (see page 5-40) to stop the preview. During the preview operation, be sure to call the `SGIdle` function (see page 5-39) frequently, so that the sequence grabber and its channels can perform the operation.
4. When you are done previewing, you can start recording or close your connection to the sequence grabber component. When you close the sequence grabber component, it automatically disposes of the channels you created.

See the sample program in Listing 5-1 on page 5-11 for an example of the preview operation.

## Recording

---

During a record operation, a sequence grabber component collects the data it captures and formats that data into a QuickTime movie. During a record operation, the sequence grabber can also play the captured data for the user. However, the sequence grabber tries to prevent the playback from interfering with the quality of the recording process.

The following paragraphs discuss the steps you must follow to record captured data.

1. As with a preview operation, your application must establish a connection to a sequence grabber component. Use the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.
2. Once you have a connection to a sequence grabber component, you must configure the component for the record operation. Use the `SGSetGWorld` function (see page 5-29) to set the graphics world in which the data is to be displayed. Allocate the appropriate channels by calling the `SGNewChannel` function (see page 5-31). You must call this function once for each channel to be used by the sequence grabber component. Use the `SGSetChannelUsage` function (see page 5-59) to specify that each channel is to be used for recording. At this time, you can specify whether the sequence grabber is to play that channel's data while recording. You can then use the appropriate channel configuration functions to prepare the channel for the record operation. For video channels, use the functions discussed in "Working With Video Channels" beginning on page 5-77. For sound channels, use the functions discussed in "Working With Sound Channels" beginning on page 5-92.
3. You must specify a movie file for use by the sequence grabber during the record operation. Use the `SGSetDataOutput` function (see page 5-26) to specify this movie file. This function also allows you to control whether the sequence grabber adds the movie resource to the movie file and whether it replaces existing data or appends the new movie to the file.
4. You can limit the amount of data that is captured during a record operation. The `SGSetMaximumRecordTime` function (see page 5-53) establishes a time limit for the record operation. The `SGSetChannelMaxFrames` function (see page 5-63) limits the number of frames of data that the sequence grabber collects from a specific channel.
5. You start the record operation by calling the `SGStartRecord` function (see page 5-38). The sequence grabber component then begins collecting data from the channels you have created, stores the data in a QuickTime movie, and, optionally, plays that data appropriately. You can pause and restart the record process by calling the `SGPause` function (see page 5-41). During the record operation, be sure to call the `SGIdle` function (see page 5-39) frequently, so that the sequence grabber and its channels can perform the operation. Use the `SGStop` function (see page 5-40) to stop recording. At this time, the sequence grabber saves the movie in your movie file, if you have chosen to do so.



## Sequence Grabber Components

6. When you are done recording, you can go back to previewing or close your connection to the sequence grabber component. When you close the sequence grabber component, it automatically disposes of the channels you created as well as any movies it has created.

## Playing Captured Data and Saving It in a QuickTime Movie

---

This section supplies a sample program that shows how to use a sequence grabber component to preview and record captured data. The program is divided into groups of functions that do the following tasks:

- initialization
- video and sound channel creation
- sequence preview
- capture of sound and video sequences
- drawing over video frames during a capture operation

## Initializing a Sequence Grabber Component

---

Listing 5-1 provides a sample function that creates and initializes a default sequence grabber component for a specified window (using the `OpenDefaultComponent` and `SGInitialize` functions, respectively). It then sets the graphics world of the sequence grabber component to the specified window with the `SGSetGWorld` function. Note that the `CloseComponent` function is called for housekeeping purposes in case the sequence grabber component fails. For more on `OpenDefaultComponent` and `CloseComponent`, see the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox*. For details on `SGInitialize` and `SGSetGWorld`, see page 5-25 and page 5-29, respectively.

**Listing 5-1**      Initializing a sequence grabber component

```
SeqGrabComponent MakeSequenceGrabber (WindowPtr aWindow)
{
    SeqGrabComponent anSG;
    OSErr err = noErr;

    /* open up the default sequence grabber */
    anSG = OpenDefaultComponent (SeqGrabComponentType, 0);
    if (anSG) {
```

## Sequence Grabber Components

```

    /* initialize the default sequence grabber component */
    err = SGInitialize (anSG);
    if (!err) {
        /* set the sequence grabber's graphics world to the
           specified window */
        err = SGSetGWorld (anSG, (CGrafPtr) aWindow, nil);
    }
}
if (err && anSG) {
    /* clean up on failure */
    CloseComponent (anSG);
    anSG = nil;
}
return anSG;
}

```

### Creating a Sound Channel and a Video Channel

---

Listing 5-2 supplies a sample function that attempts to create a video channel and a sound channel for the sequence grabber component that was created in Listing 5-1. The boundaries of the video channel are set to the specifications of the `bounds` parameter. The channel's usage is always set to allow previewing. If the value of the `willRecord` parameter is `true`, then the usage of the channel is set to allow recording also.

The `SGNewChannel` function (described on page 5-31) uses the `VideoMediaType` constant to create a video channel and the `SoundMediaType` constant to create a sound channel. The `SGSetChannelBounds` function (described on page 5-65) specifies the boundaries of the video channel. The `SGSetChannelUsage` function (described on page 5-59) specifies whether the video and the sound channels are used for preview or record operations. The `SGDisposeChannel` function (described on page 5-34) cleans up upon failure for each of the channels.

---

**Listing 5-2**      Creating a sound channel and a video channel

```

void MakeGrabChannels (SeqGrabComponent anSG,
                      SGChannel *videoChannel,
                      SGChannel *soundChannel,
                      const Rect *bounds, Boolean willRecord)
{
    OSErr err;
    long usage;
    /* figure out the usage */

```

## Sequence Grabber Components

```

usage = seqGrabPreview;          /* always previewing */
if (willRecord)
    usage |= seqGrabRecord;      /* sometimes recording */

/* create a video channel */
err = SGNewChannel (anSG, VideoMediaType, videoChannel);
if (!err) {

    /* set boundaries for new video channel */
    err = SGSetChannelBounds (*videoChannel, bounds);

    /* set usage for new video channel */
    if (!err)
        err = SGSetChannelUsage (*videoChannel,
                                   usage | seqGrabPlayDuringRecord);
    if (err) {

        /* clean up on failure */
        SGDisposeChannel (anSG, *videoChannel);
        *videoChannel = nil;
    }
}

/* create a sound channel */
err = SGNewChannel (anSG, SoundMediaType, soundChannel);
if (!err) {

    /* set usage of new sound channel */
    err = SGSetChannelUsage (*soundChannel, usage);
    if (err) {

        /* clean up on failure */
        SGDisposeChannel(anSG, *soundChannel);
        *soundChannel = nil;
    }
}
}

```

## Previewing Sound and Video Sequences in a Window

---

Listing 5-3 shows how to use the sequence grabber component to preview sound and video sequences in a window. Clicking the content area of the window causes the sequence grabber to pause until the mouse button is released.

The Image Compression Manager's `GetBestDeviceRect` function helps you determine the best monitor for the window. The `SGStartPreview` function (described on page 5-37) begins the preview of the sound and video sequences. The `SGIdle` function (described on page 5-39) grants the sequence grabber component the time it needs to preview data. The `SGUpdate` function (described on page 5-39) informs the sequence grabber of the update event. The Window Manager's `BeginUpdate` and `EndUpdate` functions respond to the event. The `SGPause` function (described on page 5-41) instructs the sequence grabber to suspend and resume its preview operation. In this example, it is used to suspend the preview operation while the mouse button is held down. Finally, the `SGStop` function (described on page 5-40) halts the action of the sequence grabber component. The Component Manager's `CloseComponent` function closes the component connection. The Window Manager's `DisposeWindow` function disposes of the window.

---

**Listing 5-3**      Previewing sound and video sequences in a window

```
void CheckError(OSErr error, Str255 displayString)
{
    if (error == noErr) return;
    if (displayString[0] > 0)
        DebugStr(displayString);
    ExitToShell();
}

Boolean IsQuickTimeInstalled (void)
{
    short    error;
    long     result;
    error = Gestalt (gestaltQuickTime, &result);
    return (error == noErr);
}

void initialize (void)
{
    OSErr err;
```

## Sequence Grabber Components

```

    InitGraf (&qd.thePort);
    InitFonts ();
    InitWindows ();
    InitMenus ();
    TEInit ();
    InitDialogs (nil);
    MaxApplZone();

    if (!IsQuickTimeInstalled())
        CheckError(-1, "\pPlease install QuickTime and try again.");

    err = EnterMovies ();
    CheckError(err, "\pUnable to initialize Movie Toolbox.");
}

WindowPtr makeWindow(void)
{
    WindowPtr aWindow;
    Rect windowRect = {0, 0, 120, 160};
    Rect bestRect;

    /* figure out the best monitor for the window */
    GetBestDeviceRect (nil, &bestRect);

    /* put the window in the top left corner of that monitor */
    OffsetRect(&windowRect, bestRect.left + 10, bestRect.top + 50);

    /* create the window */
    aWindow = NewCWindow (nil, &windowRect, "\pGrabber",
                        true, noGrowDocProc, (WindowPtr)-1,
                        true, 0);

    /* and set the port to the new window */
    SetPort(aWindow);

    return aWindow;
}

```

## Sequence Grabber Components

```

main (void)
{
    WindowPtr theWindow;
    SeqGrabComponent theSG;
    SGChannel videoChannel, soundChannel;
    Boolean done = false;
    OSErr err;

    initialize();
    theWindow = makeWindow();
    theSG = makeSequenceGrabber(theWindow);
    if (!theSG) return;

    makeGrabChannels(theSG, &videoChannel, &soundChannel,
                    &theWindow->portRect, false);
    if ((videoChannel == nil) && (soundChannel == nil))
        CheckError(-1, "\pNo sound or video available.");

    err = SGStartPreview(theSG);
    CheckError(err, "\pCan't start preview");

    while (!done) {
        AlignmentProcRecord alignProc;
        short part;
        WindowPtr whichWindow;
        EventRecord theEvent;

        GetNextEvent(everyEvent, &theEvent);

        switch (theEvent.what) {
            case nullEvent: /* give the sequence grabber time */
                err = SGIdle (theSG);
                if (err) done = true;
                break;

            case updateEvt: if (theEvent.message == (long)theWindow) {
                /* inform the sequence grabber of the
                 * update */
                SGUpdate(theSG, ((WindowPeek)
                                theWindow)->updateRgn);
                /* and swallow the update event */
                BeginUpdate(theWindow);
                EndUpdate(theWindow);
            }
        }
    }
}

```

## Sequence Grabber Components

```

        break;

        case mouseDown:part = FindWindow (theEvent.where,
                                           &whichWindow);
        if (whichWindow != theWindow) break;
        switch (part) {
            case inContent:
                /* pause until mouse button is
                   released */
                SGPause (theSG, true);
                while (StillDown())
                    ;
                SGPause(theSG, false);
                break;
            case inGoAway:
                done = TrackGoAway (theWindow,
                                    theEvent.where);

                break;
            case inDrag:
                /* pause when dragging window so video
                   doesn't draw in the wrong place */
                SGPause (theSG, true);
                SGGetAlignmentProc (theSG, &alignProc);
                DragAlignedWindow (theWindow,
                                   theEvent.where,
                                   &screenBits.bounds,
                                   nil, &alignProc);

                SGPause (theSG, false);
                break;
        }
        break;
    }
}
/* clean up */
SGStop (theSG);
CloseComponent (theSG);
DisposeWindow (theWindow);
}

```

## Capturing Sound and Video Data

---

Listing 5-4 uses the sequence grabber component to capture ten seconds of sound and video data. It prompts the user for the name of the file to create. The `SGSettingsDialog` function (described on page 5-48) is issued to invoke the default sound and video capture settings dialog boxes. These default dialog boxes allow the user to configure the settings for the capture operations. The `SGSetMaximumRecordTime` function (described on page 5-53) indicates how long the capture operations will last. The `SGStartRecord` function (described on page 5-38) specifies the time at which the capture operations will begin. The `SGIdle` function (described on page 5-39) grants the time needed to confirm the capture operations. Finally, the `SGStop` function (described on page 5-40) and the Window Manager's `DisposeWindow` routine are called in order to complete the capture of the sequences.

---

**Listing 5-4**      Capturing sound and video

```
main (void)
{
    WindowPtr theWindow;
    CGrafPort tempPort;
    SeqGrabComponent theSG;
    SGChannel videoChannel, soundChannel;
    OSErr err;

    initialize();
    theWindow = makeWindow();

    theSG = makeSequenceGrabber(theWindow);
    if (!theSG) return;
    err = setGrabFile(theSG);
    CheckError(err, "\pNo output file");

    makeGrabChannels (theSG, &videoChannel, &soundChannel,
                     &theWindow->portRect, true);
    if ((videoChannel == nil) && (soundChannel == nil))
        CheckError(-1, "\pNo sound or video available.");
}
```



## Sequence Grabber Components

```

    if (videoChannel)
        SGSettingsDialog (theSG, videoChannel, 0, nil,
                        DoTheRightThing, nil, 0);
    if (soundChannel)
        SGSettingsDialog(theSG, soundChannel, 0, nil,
                        DoTheRightThing, nil, 0);

    err = SGSetMaximumRecordTime(theSG, 10 * 60);
    CheckError(err, "\pCan't set max record time");

    err = SGStartRecord (theSG);
    CheckError(err, "\pCan't start record");

    while (!err)
        err = SGIdle (theSG);
    if (err == grabTimeComplete)
        err = noErr;
    CheckError(err, "\pError while recording");

    err = SGStop(theSG);
    CheckError(err, "\pError creating movie");

    CloseComponent(theSG);
    DisposeWindow(theWindow);
}

```

## Setting Up the Video Bottleneck Functions

Listing 5-5 shows how to set up the video bottleneck functions of the sequence grabber video channel component. For more information on the video bottleneck functions, see “Utility Functions for Video Channel Callback Functions” beginning on page 5-102. Inside the main event loop in Listing 5-4, you should add the following lines after you call the `SGSetMaximumRecordTime` function (described on page 5-53).

**Listing 5-5**      Setting up the video bottleneck functions

```

    if (videoChannel) {
        err = setupVideoBottlenecks (videoChannel, theWindow,
                                    &tempPort);
        CheckError(err, "\pCouldn't set video bottlenecks");
    }

```

## Drawing Information Over Video Frames During Capture

---

Listing 5-6 shows how to use the video bottleneck functions of the sequence grabber video channel component to draw the letters “QT” over each video frame as it is captured.

---

**Listing 5-6** Drawing information over video frames during capture

```
pascal ComponentResult myGrabFrameComplete (SGChannel c,
                                             short bufferNum,
                                             Boolean *done,
                                             long refCon)
{
    ComponentResult err;

    /* call the default grab-complete function */
    err = SGGrabFrameComplete (c, bufferNum, done);
    if (*done) {

        /* frame is done */
        CGrafPtr savePort;
        GDHandle saveGD;
        PixMapHandle bufferPM, savePM;
        Rect bufferRect;
        CGrafPtr tempPort = (CGrafPtr)refCon;

        /* set to our temporary port */
        GetGWorld (&savePort, &saveGD);
        SetGWorld (tempPort, nil);

        /* find out about this buffer */
        err = SGGetBufferInfo (c, bufferNum, &bufferPM, &bufferRect,
                               nil, nil);
        if (!err) {

            /* set up to draw into this buffer */
            savePM = tempPort->portPixMap;
            SetPortPix(bufferPM);

            /* draw some text into the buffer */
            TextMode (srcXor);
```

## Sequence Grabber Components

```

        MoveTo (bufferRect.right - 20, bufferRect.bottom - 14);
        DrawString ("\pQT");
        TextMode(srcOr);
        /* restore temporary port */
        SetPortPix (savePM);
    }
    SetGWorld (savePort, saveGD);
}
return err;
}

OSErr setupVideoBottlenecks (SGChannel videoChannel, WindowPtr w,
                             CGrafPtr tempPort)
{
    OSErr err;
    err = SGSetChannelRefCon (videoChannel, (long)tempPort);
    if (!err) {
        VideoBottles vb;

        /* get the current bottlenecks */
        vb.procCount = 9;
        err = SGGetVideoBottlenecks (videoChannel, &vb);
        if (!err) {
            /* add our GrabFrameComplete function */
            vb.grabCompleteProc = myGrabFrameComplete;
            err = SGSetVideoBottlenecks (videoChannel, &vb);

            /* set up the temporary port */
            OpenCPort (tempPort); /* create a temporary port
                                   for drawing */
            SetRectRgn (tempPort->visRgn, -32000, -32000, 32000,
                        32000); /* with a wide open visible
                                   and clip region . . . */
            CopyRgn (tempPort->visRgn, tempPort->clipRgn);
                                   /* so that you can use it in
                                   any video buffer */
            PortChanged ((GrafPtr)tempPort);
                                   /* tell QuickDraw about the
                                   changes */
        }
    }

    return err;
}

```

## Sequence Grabber Components Reference

---

This section describes the data structures and functions that are specific to sequence grabber components.

### Data Types

---

This section describes the compression information structure and the sequence grabber frame information structure.

#### Note

You only need to know about the frame information structure if you are creating a sequence grabber component. If you are not creating a sequence grabber component, you may skip this section. ♦

### The Compression Information Structure

---

The compression information structure defines the characteristics of a buffer that contains a captured image that has been compressed. Callback functions use compression information structures to exchange information about compressed images. For example, the compress-complete function must format a compression information record whenever a video frame is compressed (see “Video Channel Callback Functions” beginning on page 5-99 for more information about the compress-complete callback function). The SGCompressInfo data type defines a compression information structure.

```
struct SGCompressInfo {
    Ptr          buffer;      /* buffer for compressed image */
    unsigned long bufferSize; /* bytes of image data in buffer */
    unsigned char similarity; /* relative similarity */
    unsigned char reserved;   /* reserved--set to 0 */
};
typedef struct SGCompressInfo SGCompressInfo;
```

#### Field descriptions

|            |  |
|------------|--|
| buffer     | Points to the buffer that contains the compressed image. This pointer must contain a 32-bit clean address. |
| bufferSize | Specifies the number of bytes of image data in the buffer.   |

|            |   |
|------------|---|
| similarity | Indicates the relative similarity of this image to the previous image in a sequence. A value of 0 indicates that the current frame is a key frame in the sequence. A value of 255 indicates that the current frame is identical to the previous frame. Values from 1 through 254 indicate relative similarity, ranging from very different (1) to very similar (254). |
| reserved   | Reserved for use by Apple. Set this field to 0.   |

The Frame Information Structure

The frame information structure defines a frame for a sequence grabber component and sequence grabber channel components. The SeqGrabFrameInfo data type defines the format of a frame information structure.

```
struct SeqGrabFrameInfo {
    long      frameOffset;    /* offset to the sample */
    long      frameTime;      /* time that frame was captured */
    long      frameSize;      /* number of bytes in sample */
    SGChannel frameChannel;    /* current connection to channel */
    long      frameRefCon;     /* reference constant for channel */
};
```

Field descriptions

|              |   |
|--------------|---|
| frameOffset  | Specifies the offset to the sample.   |
| frameTime    | Specifies the time at which a sequence grabber channel component captured the frame. This time value is relative to the data sequence. That is, this time is not represented in the context of any fixed time scale. Rather, the channel component must choose and use a time scale consistently for all sample references. |
| frameSize    | Specifies the number of bytes in the sample described by the sample reference.  |
| frameChannel | Identifies the current connection to the channel component.   |
| frameRefCon  | Contains a reference constant for use by the channel component. A channel component can use this value in any way that is appropriate. For example, video channel components may use this value to store a reference to frame differencing information for a temporally compressed image sequence.                          |

## Sequence Grabber Component Functions

---

This section describes the functions that are provided by sequence grabber components. These functions are described from the perspective of an application developer. If you are developing a sequence grabber component, your component must behave as described here.

This section discusses the following groups of functions:

- “Configuring Sequence Grabber Components” describes the functions that allow you to configure a sequence grabber component, including creating channels for the component.
- “Controlling Sequence Grabber Components” discusses the functions that allow you to control a record or preview operation.
- “Working With Sequence Grabber Settings” discusses the functions that allow you to obtain sequence grabber configuration data from the user.
- “Working With Sequence Grabber Characteristics” describes functions that allow you to manage some of the detailed characteristics of a sequence grabber component.
- “Working With Channel Characteristics” describes functions that allow you to configure the general characteristics of a sequence grabber channel.
- “Working With Channel Devices” discusses functions that allow you to determine the device that is attached to a sequence grabber channel.
- “Working With Video Channels” describes functions that allow you to configure video channels.
- “Working With Sound Channels” discusses functions that allow you to configure sound channels.
- “Video Channel Callback Functions” describes the callback functions that are supported by video channels.
- “Utility Functions for Video Channel Callback Functions” discusses a number of utility functions that sequence grabber components provide for use by callback functions.

## Configuring Sequence Grabber Components

---

Sequence grabber components provide a number of functions that allow you to establish the environment for grabbing or previewing digitized data. Before you can start a record or a preview operation, you must initialize the sequence grabber component, establish the channels that will be used, define the display environment for the operation, and determine the optimum screen position for the sequence grabber. In addition, if you are performing a record operation, you must define a destination movie file. This section describes the sequence grabber component functions that allow you to perform these tasks.

You can use the `SGInitialize` function to initialize a sequence grabber component. Before you can call this function, you must establish a connection to the sequence

## Sequence Grabber Components

grabber by calling the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

The `SGNewChannel` function allows you to create channels for the sequence grabber for an operation. You can use the `SGNewChannelFromComponent` function to create a new channel using a specified channel component. Use the `SGDisposeChannel` function to dispose of those channels that you are no longer using.

You can use the `SGGetIndChannel` function to retrieve information about the channels that are currently in use by the sequence grabber.

You can use the `SGSetGWorld` and `SGGetGWorld` functions to establish the display environment for the sequence grabber. These functions affect only those channels that work with data that has visual information.

The `SGSetDataOutput` and `SGGetDataOutput` functions allow you to identify the movie file that is currently assigned to the sequence grabber. You only use these functions when you are performing a record operation.

The `SGSetDataProc` function allows you to assign a data function to a channel. The sequence grabber calls your data function whenever it writes movie data to the output file.

The `SGGetAlignmentProc` function allows you to determine a sequence grabber's optimum screen position to ensure the best performance and appearance.

## SGInitialize

---

The `SGInitialize` function allows you to initialize the sequence grabber component. Before you can call this function you must establish a connection to the sequence grabber component. Use the Component Manager's `OpenDefaultComponent` or `OpenComponent` function to establish a component connection.

```
pascal ComponentResult SGInitialize (SeqGrabComponent s);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

### DESCRIPTION

You must call the `SGInitialize` function before you call any other sequence grabber component functions. If this function returns a nonzero result code, you should close your connection to the sequence grabber component.

### RESULT CODES

Memory Manager errors

## SGSetDataOutput

---

The `SGSetDataOutput` function allows you to specify the movie file for a record operation and to specify other options that govern the operation. The sequence grabber component stores the data that is obtained during the record operation as a QuickTime movie in this file. This function also allows you to control some aspects of the record operation, which are related to output, by specifying control flags. These flags are discussed in the function description that follows.

```
pascal ComponentResult SGSetDataOutput (SeqGrabComponent s,
                                         FSSpec *movieFile,
                                         long whereFlags);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**movieFile** Contains a pointer to the movie file for this record operation.

**whereFlags** Contains flags that control the record operation. The following flags are defined by the `SeqGrabDataOutputEnum` data type; you must set either the `seqGrabToDisk` flag or the `seqGrabToMemory` flag to 1 (set unused flags to 0).

`seqGrabToDisk`

Instructs the sequence grabber component to write the recorded data to a QuickTime movie in the movie file specified by the `movieFile` parameter. If you set this flag to 1, the sequence grabber writes the data to the file as the data is recorded. Set this flag to 0 if you set the `seqGrabToMemory` flag to 1 (only one of these two flags may be set to 1).

`seqGrabToMemory`

Instructs the sequence grabber component to store the recorded data in memory until the recording process is complete. The sequence grabber then writes the recorded data to the movie file specified by the `movieFile` parameter. This technique provides better performance than recording directly to the movie file, but it limits the amount of data you can record. Set this flag to 1 to record to memory. Set this flag to 0 if you set the `seqGrabToDisk` flag to 1 (only one of these two flags may be set to 1).

`seqGrabDontUseTempMemory`

Prevents the sequence grabber component from using temporary memory during the record operation. By default, the sequence grabber component and its channel components use as much temporary memory as necessary



to perform the record operation. Set this flag to 1 to prevent the sequence grabber component and its channel components from using temporary memory.

`seqGrabAppendToFile`

Directs the sequence grabber component to add the recorded data to the data fork of the movie file specified by the `movieFile` parameter. By default, the sequence grabber component deletes the movie file and creates a new file containing one movie and the corresponding movie resource. Set this flag to 1 to cause the sequence grabber component to append the recorded data to the data fork of the movie file and create a new movie resource in that file.

`seqGrabDontAddMovieResource`

Prevents the sequence grabber component from adding the new movie resource to the movie file specified by the `movieFile` parameter. By default, the sequence grabber component creates a new movie resource and adds that resource to the movie file. Set this flag to 1 to prevent the sequence grabber component from adding the movie resource to the movie file. You are then responsible for adding the resource to a file, if you so desire.

`seqGrabDontMakeMovie`

Prevents the sequence grabber component from creating a movie. By default, the sequence grabber component creates a new movie resource and adds the captured data to that movie. If you set this flag to 1, the sequence grabber still calls your data function, but does not write any data to the movie file.

DESCRIPTION

If you are performing a preview operation, you do not need to use the `SGSetDataOutput` function.

RESULT CODES

|                                       |       |  |
|---------------------------------------|-------|--|
| <code>notEnoughMemoryToGrab</code>    | -9403 | Insufficient memory for record operation     |
| <code>notEnoughDiskSpaceToGrab</code> | -9404 | Insufficient disk space for record operation |
| File Manager errors                   |       |  |
| Memory Manager errors                 |       |  |

## SGGetDataOutput

---

The `SGGetDataOutput` function allows you to determine the movie file that is currently assigned to a sequence grabber component and the control flags that would govern a record operation.

```
pascal ComponentResult SGGetDataOutput (SeqGrabComponent s,
                                       FSSpec *movieFile,
                                       long *whereFlags);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**movieFile** Contains a pointer to a file system specification record that is to receive information about the movie file for this record operation.

**whereFlags** Contains a pointer to a long integer that is to receive flags that control the record operation. The following flags are defined (unused flags are set to 0):

`seqGrabToDisk`

Instructs the sequence grabber component to write the recorded data to a QuickTime movie in the movie file specified by the `movieFile` parameter. If this flag is set to 1, the sequence grabber writes the data to the file as the data is recorded.

`seqGrabToMemory`

Instructs the sequence grabber component to store the recorded data in memory until the recording process is complete. The sequence grabber then writes the recorded data to the movie file specified by the `movieFile` parameter. This technique provides better performance than recording directly to the movie file, but it limits the amount of data you can record. If this flag is set to 1, the sequence grabber component is recording to memory.

`seqGrabDontUseTempMemory`

Prevents the sequence grabber component from using temporary memory during the record operation. By default, the sequence grabber component and its channel components use as much temporary memory as necessary to perform the record operation. If this flag is set to 1, the sequence grabber component and its channel components do not use temporary memory.

`seqGrabAppendToFile`

Directs the sequence grabber component to add the recorded data to the data fork of the movie file specified by the `movieFile` parameter. By default, the sequence grabber component deletes the movie file and creates a

new file containing one movie and its movie resource. If this flag is set to 1, the sequence grabber component appends the recorded data to the data fork of the movie file and creates a new movie resource in that file.

`seqGrabDontAddMovieResource`

Prevents the sequence grabber component from adding the new movie resource to the movie file specified by the `movieFile` parameter. By default, the sequence grabber component creates a new movie resource and adds that resource to the movie file. If this flag is set to 1, the sequence grabber component does not add the movie resource to the movie file. You are then responsible for adding the resource to a file, if you so desire.

`seqGrabDontMakeMovie`

Prevents the sequence grabber component from creating a movie. By default, the sequence grabber component creates a new movie resource and adds the captured data to that movie. If this flag is set to 1, the sequence grabber still calls your data function, but does not write any data to the movie file.

DESCRIPTION

You set these characteristics by calling the `SGSetDataOutput` function, which is described in the previous section. If you have not set these characteristics before calling the `SGGetDataOutput` function, the returned data is meaningless.

SGSetGWorld

The `SGSetGWorld` function allows you to establish the graphics port and device for a sequence grabber component. The sequence grabber component displays the recorded or previewed data in this graphics world.

```
pascal ComponentResult SGSetGWorld (SeqGrabComponent s,
                                     CGrafPtr gp, GDHandle gd);
```

- `s` Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.
- `gp` Specifies the destination graphics port. The specified graphics port must be a color graphics port. Set this parameter to `nil` to use the current graphics port.
- `gd` Specifies the destination graphics device. Set this parameter to `nil` to use the current device. If the `gp` parameter specifies a graphics world, set this parameter to `nil` to use that graphics world's graphics device.

## Sequence Grabber Components

**DESCRIPTION**

You must call this function if you are working with any channels that collect visual data. If you are working only with data that has no visual representation, you do not need to call this function. The sequence grabber component performs this operation implicitly when you call the `SGInitialize` function (described on page 5-25), and the component uses your application's current graphics port.

You cannot call this function during a record or preview operation or after you have prepared the sequence grabber component for a record or preview operation (by calling the `SGPrepare` function, which is described on page 5-43).

**IMPORTANT**

The window in which the sequence grabber is to draw video frames as defined by `SGSetGWorld` must be visible before you call the `SGPrepare` function. Otherwise, the sequence grabber does not display the frames properly. For details, see the discussion of `SGPrepare` beginning on page 5-43. ▲

**RESULT CODE**

`cantDoThatInCurrentMode`      -9402      Request invalid in current mode

**SGGetGWorld**

---

The `SGGetGWorld` function allows you to determine the graphics port and device for a sequence grabber component.

```
pascal ComponentResult SGGetGWorld (SeqGrabComponent s,
                                     CGrafPtr *gp, GDHandle *gd);
```

|                 |  |
|-----------------|--|
| <code>s</code>  | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| <code>gp</code> | Contains a pointer to a location that is to receive a pointer to the destination graphics port. Set this parameter to <code>nil</code> if you are not interested in this information.  |
| <code>gd</code> | Contains a pointer to a location that is to receive a handle to the destination graphics device. Set this parameter to <code>nil</code> if you are not interested in this information.   |

DESCRIPTION

The sequence grabber component displays the recorded or previewed data in this graphics world.

SEE ALSO

You can establish the graphics port and device for a sequence grabber component by calling the `SGSetGWorld` function, which is described in the previous section.

SGNewChannel

The `SGNewChannel` function creates a sequence grabber channel and assigns a channel component to the channel. The channel component is responsible for providing digitized data to the sequence grabber component. You specify the type of channel component to be added to the sequence grabber component.

```
pascal ComponentResult SGNewChannel (SeqGrabComponent s,
                                     OSType channelType,
                                     SGChannel *ref);
```

- s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.
- channelType** Specifies the type of channel to open. This value corresponds to the component subtype value of the channel component. The following values are valid:
  - `VideoMediaType`  
Video channel
  - `SoundMediaType`  
Sound channel
- ref** Contains a pointer to the `frameChannel` field in the sequence grabber information structure that is to receive a reference to the channel that is added to the sequence grabber component. If the sequence grabber component successfully locates and connects to an appropriate channel component, the sequence grabber component returns a reference to the channel component into the field referred to by this parameter. If the sequence grabber component cannot open a connection, it sets the result code to a nonzero value.

DESCRIPTION

The sequence grabber component locates, and attempts to connect to, an appropriate channel component. If the sequence grabber component cannot locate or connect to a channel component, it returns a nonzero result code.

## Sequence Grabber Components

## RESULT CODES

`couldntGetRequiredComponent`    **-9405**    Component not found  
 Memory Manager errors

## SEE ALSO

When you are done with the sequence grabber component, you can dispose of the channels you have used by calling the `SGDisposeChannel` function, which is described on page 5-34. However, when you close the sequence grabber component, it automatically disposes of all its channels, so this function is usually unnecessary.

If you want to use a specific channel component, you may use the `SGNewChannelFromComponent` function, which is described next.

## SGNewChannelFromComponent

---

The `SGNewChannelFromComponent` function creates a sequence grabber channel and assigns a channel component to the channel. The channel component is responsible for providing digitized data to the sequence grabber component. You specify the channel component to be used.

```
pascal ComponentResult SGNewChannelFromComponent
    (SeqGrabComponent s, SGChannel *newChannel,
     Component sgChannelComponent);
```

**s**                      Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**newChannel**           Contains a pointer to a channel component that is to receive a reference to the channel that is added to the sequence grabber component. If the sequence grabber component successfully locates and connects to the specified channel component, the sequence grabber component returns a reference to the channel component into the field referred to by this parameter. If the sequence grabber component cannot open a connection, it sets the result code to a nonzero value.

**sgChannelComponent**   Identifies the channel component to use. You supply a component ID value to the sequence grabber. The sequence grabber then opens a connection to that channel component and returns your connection ID in the field specified by the `newChannel` parameter. You may obtain a component ID value by calling the Component Manager's `FindNextComponent` function.

DESCRIPTION

The sequence grabber component locates and connects to the specified channel component. If the sequence grabber component cannot locate or connect to the channel component, it returns a nonzero result code.

This function is similar to the `SGNewChannel` function, except that this function allows you to specify a particular component rather than just a component subtype value. Use this function if you want to connect to a specific component.

RESULT CODES

`couldntGetRequiredComponent`     `-9405`     Component not found  
Memory Manager errors

SEE ALSO

You may also use the `SGNewChannel` function to establish a new channel. That function requires only a component subtype value, and is described on page 5-31.

When you are done with the sequence grabber component, you can dispose of the channels you have used by calling the `SGDisposeChannel` function, which is described on page 5-34.

SGGetIndChannel

The `SGGetIndChannel` function allows you to collect information about all of the channel components currently in use by a sequence grabber component.

```
pascal ComponentResult SGGetIndChannel (SeqGrabComponent s,
                                         short index,
                                         SGChannel *ref,
                                         OSType *chanType);
```

- `s` Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.
- `index` Specifies an index value. This value identifies the channel to be queried. The first channel has an index value of 1.
- `ref` Contains a pointer to a field to receive a value identifying your connection to the channel. If you do not want to receive this information, set this parameter to `nil`.

## Sequence Grabber Components

**chanType**      Contains a pointer to a field to receive the channel's subtype value. This value indicates the media type supported by the channel component. The following values are valid:

VideoMediaType  
                    Video channel

SoundMediaType  
                    Sound channel

If you do not want to receive this information, set this parameter to `nil`.

**DESCRIPTION**

You may use the `SGGetIndChannel` function to retrieve information about each of the channel components currently in use by a sequence grabber component. You identify the channel in which you are interested by specifying an index value. These index values start at 1 and increase sequentially; each channel has its own index value.

**RESULT CODE**

`paramErr`      -50      Component not found

**SGDisposeChannel**

---

The `SGDisposeChannel` function removes a channel from a sequence grabber component.

```
pascal ComponentResult SGDisposeChannel
                               (SeqGrabComponent s, SGChannel c);
```

**s**                Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**c**                Specifies the reference that identifies the channel you want to close. You obtain this reference from the `SGNewChannel` function, described in the previous section.

**DESCRIPTION**

You can use this function to remove a channel that you are no longer using. However, you cannot dispose of a channel that is currently active—if you are recording or previewing data, this function returns a nonzero result code.



**RESULT CODE**

badSGChannel      -9406      Invalid channel specified

**SEE ALSO**

The sequence grabber component automatically disposes of any open channels when you close your connection to the component, so you do not need to call this function prior to calling the Component Manager's `CloseComponent` function.

**SGSetDataProc**

---

The `SGSetDataProc` function allows you to specify a data function for use by the sequence grabber. Whenever any channel assigned to the sequence grabber writes data, your data function is called as well. Your data function may then write the data to another destination.

```
pascal ComponentResult SGSetDataProc (SeqGrabComponent sg,
                                       SGDataProc proc,
                                       long refCon);
```

|                     |  |
|---------------------|--|
| <code>sg</code>     | Identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function.   |
| <code>proc</code>   | Contains a pointer to your data function. To remove your data function, set this parameter to <code>nil</code> . The interface that your data function must support is described in "Application-Defined Functions" beginning on page 5-111. |
| <code>refCon</code> | Contains a reference constant. The sequence grabber provides this value to your data function.   |

**DESCRIPTION**

Your application may use the `SGSetDataProc` function to assign a data function to a sequence grabber. The sequence grabber calls your data function whenever any channel component writes data to the destination movie. You may use your data function to store the digitized data in some format other than a QuickTime movie.

**SEE ALSO**

You can instruct the sequence grabber not to write its data to a QuickTime movie by calling the `SGSetDataOutput` function and setting the `seqGrabDontMakeMovie` flag to 1. This can save processing time in cases where you do not want to create a movie. This function is discussed beginning on page 5-26.

## SGGetAlignmentProc

---

The `SGGetAlignmentProc` function allows you to obtain information about the best screen positions for a sequence grabber's video image in terms of appearance and maximum performance.

```
pascal ComponentResult SGGetAlignmentProc (SeqGrabComponent s,
                                           AlignmentProcRecordPtr alignmentProc);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**alignmentProc** Contains a pointer to an Image Compression Manager alignment function structure. The sequence grabber places its alignment information into this structure.

### DESCRIPTION

You may use the `SGGetAlignmentProc` function to retrieve information about the best screen positions for the sequence grabber's window. The sequence grabber returns information that can be used by the Image Compression Manager's alignment functions (see the chapter "Image Compression Manager" in *Inside Macintosh: QuickTime* for more information about these functions). By using this alignment information, you can place the sequence grabber's window in a position that allows for optimal display performance.

## Controlling Sequence Grabber Components

---

Sequence grabber components provide a full set of functions that allow your application to control the preview or record operation. You can use these functions to start and stop the operation, to pause data collection, and to retrieve a reference to the movie that is created during a record operation. This section describes these functions.

Use the `SGStartPreview` function to start a preview operation. The `SGStartRecord` function lets you start a record operation. The `SGStop` function allows you to stop a sequence grabber component.

You can instruct the sequence grabber to pause by calling the `SGPause` function. You can determine whether the sequence grabber is paused by calling the `SGGetPause` function.

You grant processing time to the sequence grabber by calling the `SGIdle` function. Be sure to call this function often during record and preview operations. If your application receives an update event during a record or preview operation, you should call the `SGUpdate` function.

You can prepare the sequence grabber for an upcoming preview or record operation by calling the `SGPrepare` function. This function also allows the sequence grabber to verify that it can support the parameters you have specified. By verifying the parameters you

want to use, you can improve the startup of preview and record operations. Use the `SGRelease` function to release system resources after calling the `SGPrepare` function.

You can retrieve a reference to the movie created by a record operation by calling the `SGGetMovie` function. You can determine the resource ID value assigned to the last movie resource created by the sequence grabber by calling the `SGGetLastMovieResID` function.

You can extract a picture from the video source data by calling the `SGGrabPict` function.

**SGStartPreview**

The `SGStartPreview` function instructs the sequence grabber to begin processing data from its channels.

```
pascal ComponentResult SGStartPreview (SeqGrabComponent s);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**DESCRIPTION**

The sequence grabber immediately presents the data to the user in the appropriate format, according to the channel configuration parameters you have specified (see “Working With Channel Characteristics” beginning on page 5-58 for information about configuring channels). Video data is displayed in the destination display region; sound data is played at the specified volume settings.

**RESULT CODES**

|                                      |       |                                 |
|--------------------------------------|-------|---------------------------------|
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode |
| <code>deviceCantMeetRequest</code>   | -9408 | Device cannot support grabber   |
| File Manager errors                  |       |                                 |
| Memory Manager errors                |       |                                 |

**SEE ALSO**

You stop the preview process by calling the `SGStop` function, which is described on page 5-40.

In preview mode, the sequence grabber does not save any of the data it gathers from its channels. If you want to record the data, use record mode. You start a record operation by calling the `SGStartRecord` function, which is described in the next section.

## SGStartRecord

---

The `SGStartRecord` function instructs the sequence grabber component to begin collecting data from its channels.

```
pascal ComponentResult SGStartRecord (SeqGrabComponent s);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

### DESCRIPTION

The sequence grabber stores the collected data according to the recording parameters you specify with the `SGSetDataOutput` function, which is described on page 5-26. Before calling this function, you must correctly configure the sequence grabber's channels—see “Working With Channel Characteristics” beginning on page 5-58 for information about configuring sequence grabber channels.

### RESULT CODES

|                                       |       |  |
|---------------------------------------|-------|--|
| <code>cantDoThatInCurrentMode</code>  | -9402 | Request invalid in current mode              |
| <code>notEnoughMemoryToGrab</code>    | -9403 | Insufficient memory for record operation     |
| <code>notEnoughDiskSpaceToGrab</code> | -9404 | Insufficient disk space for record operation |
| <code>deviceCantMeetRequest</code>    | -9408 | Device cannot support grabber                |
| File Manager errors                   |       |  |
| Memory Manager errors                 |       |  |

### SEE ALSO

You can switch from previewing to recording by calling this function during a preview operation—you need not stop the preview operation first. You stop the recording process by calling the `SGStop` function, which is described on page 5-40.

You can cause the sequence grabber to display the data it obtains from its channels without storing any of the data by calling the `SGStartPreview` function, which is described in the previous section.

## SGIdle

---

The `SGIdle` function provides processing time to the sequence grabber component and its channel components. After starting a preview or record operation, you should call this function as often as possible, until you stop the operation by calling `SGStop`.

### ▲ WARNING

If you do not call `SGIdle` frequently enough, you may lose data. ▲

```
pascal ComponentResult SGIdle (SeqGrabComponent s);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

### DESCRIPTION

The `SGIdle` function reports several status and error conditions by means of its result code. If you have established a time limit for a record operation by calling the `SGSetMaximumRecordTime` function (described on page 5-53), `SGIdle` returns a result code of `grabTimeComplete` when the time limit expires. In addition, `SGIdle` reports errors that are specific to the channels that are active for a given operation. If `SGIdle` returns a nonzero result code during a record operation, you should still call the `SGStop` function (described on page 5-40) so that the sequence grabber can store the data it has collected.

### RESULT CODES

|                                      |       |                                       |
|--------------------------------------|-------|---------------------------------------|
| <code>grabTimeComplete</code>        | -9401 | Time for record operation has expired |
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode       |
| File Manager errors                  |       |                                       |
| Memory Manager errors                |       |                                       |

## SGUpdate

---

The `SGUpdate` function allows you to tell the sequence grabber that it must refresh its display.

```
pascal ComponentResult SGUpdate (SeqGrabComponent s,
                                RgnHandle updateRgn);
```

## Sequence Grabber Components

- s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.
- updateRgn** Indicates the part of the window that has been changed. You may use this parameter to specify a portion of the window that you know has been changed. You can obtain this information by examining the appropriate window record. For example:

```
SGUpdate (theSG, ((WindowPeek)updateWindow)->updateRgn);
```

If you set this parameter to `nil`, the sequence grabber uses the window's current visible region.

**DESCRIPTION**

You may use the `SGUpdate` function to tell the sequence grabber that it must refresh its display. You should call this function whenever you receive an update event for a window that contains a sequence grabber display. You should call this function before calling the Window Manager's `BeginUpdate` function.

Your application should avoid drawing where the sequence grabber is displaying video. Doing so may cause some video digitizer components to stop displaying video.

**SPECIAL CONSIDERATIONS**

It is dangerous to allow an update event to occur during recording. Many digitizers capture directly to the screen; thus, an update event will result in data loss.

**RESULT CODES**

|                                    |       |                               |
|------------------------------------|-------|-------------------------------|
| <code>paramErr</code>              | -50   | Component not found           |
| <code>deviceCantMeetRequest</code> | -9408 | Device cannot support grabber |

**SGStop**

---

The `SGStop` function stops a preview or record operation.

```
pascal ComponentResult SGStop (SeqGrabComponent s);
```

- s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**DESCRIPTION**

The sequence grabber releases any system resources it used during the operation, such as temporary memory. In the case of a record operation, the sequence grabber stores the

collected movie data in the assigned movie file—you specify the movie file by calling the `SGSetDataOutput` function, which is described on page 5-26.

RESULT CODES

`cantDoThatInCurrentMode`     -9402     Request invalid in current mode  
File Manager errors  
Memory Manager errors

SGPause

You can suspend or restart a record or preview operation by calling the `SGPause` function. You supply a byte value that instructs the sequence grabber whether to pause or restart the current operation.

```
pascal ComponentResult SGPause (SeqGrabComponent s,  
                                Byte pause);
```

- `s` Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.
- `pause` Instructs the sequence grabber whether to suspend or restart the current operation. The following values are valid:
  - `seqGrabUnpause`  
Restarts the current operation.
  - `seqGrabPause`  
Pauses the current operation.
  - `seqGrabPauseForMenu`  
Pauses the current operation so that you may display a menu. Use this option only in preview mode, just before you call the Menu Manager's `MenuSelect` or `PopUpMenuSelect` function. In this case, the sequence grabber may not pause all channels, depending upon the ability of the sequence grabber to play with acceptable quality. For example, sound channels may continue to play while video channels are paused.

DESCRIPTION

The `SGPause` function does not release any system resources or temporary memory associated with the current operation. Consequently, it is generally much faster than using the `SGStop` and `SGStartRecord` functions or the `SGStartPreview` function to suspend an operation.

## Sequence Grabber Components

## SPECIAL CONSIDERATIONS

When you restart the operation, the sequence grabber component may be unable to satisfy your request. This can occur, for example, if the user has moved the display window to a location that the digitizing hardware cannot support.

## RESULT CODES

|                                      |       |  |
|--------------------------------------|-------|--|
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode          |
| <code>notEnoughMemoryToGrab</code>   | -9403 | Insufficient memory for record operation |
| <code>deviceCantMeetRequest</code>   | -9408 | Device cannot support grabber            |

File Manager errors

Memory Manager errors

## SEE ALSO

You may determine whether the sequence grabber is paused by calling the `SGGetPause` function, which is described next.

**SGGetPause**

---

You can determine whether the sequence grabber is paused by calling the `SGGetPause` function.

```
pascal ComponentResult SGGetPause (SeqGrabComponent s,
                                   Byte *paused);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**paused** Contains a pointer to a field that is to receive a value that indicates whether the sequence grabber is currently paused. The following values are valid:

`seqGrabUnpause`

The sequence grabber is not paused.

`seqGrabPause`

The sequence grabber is paused—all channels are stopped.

`seqGrabPauseForMenu`

The sequence grabber is paused in order to display a menu—some or all of the channels may be stopped.

## DESCRIPTION

The `SGGetPause` function allows you to determine whether the sequence grabber is paused.



**SEE ALSO**

You may pause or restart the sequence grabber by calling the `SGPause` function, which is described in the previous section.

**SGPrepare**

The `SGPrepare` function instructs the sequence grabber to get ready to begin a preview or record operation (or to commence both operations). You specify the operations.

```
pascal ComponentResult SGPrepare (SeqGrabComponent s,
                                   Boolean prepareForPreview,
                                   Boolean prepareForRecord);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**prepareForPreview**

Instructs the sequence grabber component to prepare for a preview operation. Set this parameter to `true` to prepare for a preview operation. You may set both the `prepareForPreview` and `prepareForRecord` parameters to `true`.

**prepareForRecord**

Instructs the sequence grabber component to prepare for a record operation. Set this parameter to `true` to prepare for a record operation. You may set both the `prepareForPreview` and `prepareForRecord` parameters to `true`.

**DESCRIPTION**

The sequence grabber component does whatever is necessary to get ready to start the preview or record operation. This may involve allocating memory, readying hardware, and notifying the sequence grabber's channels. By calling this function, you ensure that the `SGStartRecord` or `SGStartPreview` function starts as quickly as possible.

If you do not call this function before starting a record or preview operation, the sequence grabber component makes these preparations when you start the operation. You cannot call this function after you start a preview or record operation.

If you call `SGPrepare` without subsequently starting a record or preview operation, you should call the `SGRelease` function (described in the next section). This allows the sequence grabber component to release any system resources it allocated when you called `SGPrepare`.

**SPECIAL CONSIDERATIONS**

The window in which the sequence grabber is to draw video frames (as defined by the `SGSetGWorld` function, described on page 5-29) must be visible before you call the `SGPrepare` function. Otherwise, the sequence grabber does not display the frames properly. If the window isn't visible and `SGPrepare` is called with the `prepareForPreview` parameter set to `true` and the `prepareForRecord` parameter set to `false`, and the window is subsequently shown via the Window Manager's `ShowWindow` routine, the sequence grabber won't display frames properly in the video window. The visible region of the window wasn't valid when the `SGPrepare` call was made.

**RESULT CODES**

|                                       |       |  |
|---------------------------------------|-------|--|
| <code>paramErr</code>                 | -50   | Invalid parameter specified                  |
| <code>cantDoThatInCurrentMode</code>  | -9402 | Request invalid in current mode              |
| <code>notEnoughMemoryToGrab</code>    | -9403 | Insufficient memory for record operation     |
| <code>notEnoughDiskSpaceToGrab</code> | -9404 | Insufficient disk space for record operation |
| <code>deviceCantMeetRequest</code>    | -9408 | Device cannot support grabber                |
| File Manager errors                   |       |  |
| Memory Manager errors                 |       |  |

**SGRelease**

---

The `SGRelease` function instructs the sequence grabber to release any system resources it allocated when you called the `SGPrepare` function, which is described in the previous section. You should call `SGRelease` whenever you call `SGPrepare` without subsequently starting a record or preview operation.

```
pascal ComponentResult SGRelease (SeqGrabComponent s);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

**DESCRIPTION**

When you stop a record or preview operation by calling the `SGStop` function, the sequence grabber component automatically releases the resources it uses during the operation. Consequently, you do not have to call this function after a record or preview operation.

You cannot call the `SGRelease` function during a record or preview operation.

## SGGetMovie

---

The `SGGetMovie` function returns a reference to the movie that contains the data collected during a record operation. You can use this movie identifier with Movie Toolbox functions. However, you should not dispose of this movie—it is owned by the sequence grabber component. Furthermore, the sequence grabber component disposes of this movie when you prepare for or start the next record or preview operation, or when you close the connection to the sequence grabber. If you want to work with a movie containing the collected data, use the Movie Toolbox's `NewMovieFromFile` function (see the chapter “Movie Toolbox” in *Inside Macintosh: QuickTime* for more information).

You can call this function only after you have stopped the record operation by calling the `SGStop` function.

```
pascal Movie SGGetMovie (SeqGrabComponent s);
```

**s** Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.

### DESCRIPTION

The `SGGetMovie` function returns a reference to the movie that contains the data collected during a record operation. If there is no current movie, either because you are in preview mode or because you have not yet stopped the record operation, the sequence grabber component sets this returned reference to `nil`.

### RESULT CODE

|                                      |                    |   |
|--------------------------------------|--------------------|---|
| <code>seqGrabInfoNotAvailable</code> | <code>-9407</code> | Sequence grabber cannot support request |
|--------------------------------------|--------------------|---|

## SGGetLastMovieResID

---

The `SGGetLastMovieResID` allows you to retrieve the last resource ID used by the sequence grabber component. The sequence grabber component assigns a new resource ID to each movie resource it creates. The sequence grabber creates the movie resource when you stop a record operation by calling the `SGStop` function, unless you have instructed the sequence grabber not to add the movie resource to the movie file (see the description of the `SGSetDataOutput` function beginning on page 5-26 for more information).

```
pascal ComponentResult SGGetLastMovieResID (SeqGrabComponent s,
                                             short *resID) ;
```

## Sequence Grabber Components

|                    |  |
|--------------------|--|
| <code>s</code>     | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| <code>resID</code> | Contains a pointer to an integer that is to receive the resource ID the sequence grabber assigned to the movie resource it just created.   |

## DESCRIPTION

If you want this information, you should call this function before you prepare for or start another record or preview operation—because the sequence grabber component resets this value when you start the next operation.

## RESULT CODE

`seqGrabInfoNotAvailable`    -9407    Sequence grabber cannot support request

**SGGrabPict**

The `SGGrabPict` function provides a simple interface that allows your application to obtain a QuickDraw picture from a sequence grabber component. The sequence grabber can display the picture directly, or it can write the picture to an offscreen buffer. This function is limited in scope, however, and does not allow you to control all of the parameters that govern the operation. When you call this function, the sequence grabber component obtains and configures appropriate sequence grabber channel components (if necessary), grabs the data, and then releases any components it obtained.

```
pascal ComponentResult SGGrabPict (SeqGrabComponent s,
                                   PicHandle *p,
                                   const Rect *bounds,
                                   short offscreenDepth,
                                   long grabPictFlags);
```

|                             |   |
|-----------------------------|---|
| <code>s</code>              | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function.  |
| <code>p</code>              | Contains a pointer to a field that is to receive a handle to the picture. If the <code>SGGrabPict</code> function cannot create the picture, it sets this handle to <code>nil</code> .  |
| <code>bounds</code>         | Contains a pointer to the boundary region for the picture. By default, this rectangle lies in the current graphics port. If you set the <code>grabPictOffScreen</code> flag in the <code>grabPictFlags</code> parameter to 1, the sequence grabber places the picture in an offscreen graphics world. In this case, the rectangle is interpreted in that offscreen world. |
| <code>offscreenDepth</code> | Specifies the pixel depth for the offscreen graphics world. This parameter is typically set to 0, which chooses the best available depth. If you set the  |

## Sequence Grabber Components

`grabPictOffScreen` flag in the `grabPictFlags` parameter to 1, the sequence grabber places the picture in an offscreen graphics world. You specify the pixel depth of this offscreen graphics world with this parameter. If you are displaying the picture, this parameter is ignored.

`grabPictFlags`

Contains flags that control the operation. The following flags are defined (set unused flags to 0):

`grabPictOffScreen`

Instructs the sequence grabber to place the picture in an offscreen graphics world. Set this flag to 1 to use an offscreen graphics world. In this case, you use the `offscreenDepth` parameter to specify the pixel depth in the offscreen buffer. In addition, the rectangle specified by the `bounds` parameter is applied to the offscreen buffer.

`grabPictIgnoreClip`

Instructs the sequence grabber to ignore any clipping regions you may have defined for the sequence grabber's channels. Set this flag to 1 to have the sequence grabber ignore these clipping regions.

**DESCRIPTION**

If you have created any channels for the sequence grabber component, the `SGGrabPict` function uses those channels to obtain the data for the captured image.

**SPECIAL CONSIDERATIONS**

Some digitizer sources do not support grabbing offscreen, so the `SGGrabPict` function may fail. In this case, try again grabbing onscreen.

**RESULT CODES**

|                                    |       |  |
|------------------------------------|-------|--|
| <code>notEnoughMemoryToGrab</code> | -9403 | Insufficient memory for record operation |
| <code>deviceCantMeetRequest</code> | -9408 | Device cannot support grabber            |

File Manager errors

Memory Manager errors

## Working With Sequence Grabber Settings

---

Sequence grabber components can work with channel components and panel components to collect configuration settings from the user. The functions discussed in this section allow you to direct the sequence grabber to display its settings dialog box to the user and to work with the configuration of each of the grabber's channels. See "About Sequence Grabber Components" on page 5-3 for more information about the relationship between the sequence grabber and panel components.

## Sequence Grabber Components

Use the `SGSettingsDialog` function to instruct the sequence grabber to display its settings dialog box to the user.

The `SGSetSettings` and `SGGetSettings` functions allow you to retrieve or set the sequence grabber's configuration.

The `SGSetChannelSettings` and `SGGetChannelSettings` functions work with the configuration of an individual channel.

## SGSettingsDialog

---

You may cause the sequence grabber to display its settings dialog box to the user by calling the `SGSettingsDialog` function. The user can use this dialog box to specify the configuration of a sequence grabber channel.

```
pascal ComponentResult SGSettingsDialog (SeqGrabComponent s,
                                         SGChannel c, short numPanels,
                                         Component *panelList, long flags,
                                         SGModalFilterProcPtr proc, long procRefNum);
```

|                         |   |
|-------------------------|---|
| <code>s</code>          | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function.  |
| <code>c</code>          | Identifies the channel to be configured. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, discussed on page 5-31 and page 5-32, respectively.  |
| <code>numPanels</code>  | Specifies the number of panel components to be listed in the panel component pop-up menu. You specify the panel components with the <code>panelList</code> parameter. You may use these parameters to limit the user's choice of panel components. If you set this parameter to 0 and the <code>panelList</code> parameter to <code>nil</code> , the sequence grabber lists all available panel components. |
| <code>panelList</code>  | Contains a pointer to an array of component identifiers. The sequence grabber presents only these components in the panel component pop-up menu. You specify the number of identifiers in the array with the <code>numPanels</code> parameter. If you set this parameter to <code>nil</code> , the sequence grabber lists all available panel components.   |
| <code>flags</code>      | Reserved for Apple Computer. Set this parameter to 0.   |
| <code>proc</code>       | Specifies an event filter function. Because the sequence grabber's settings dialog box is a movable modal dialog box, you must supply an event filter function to process update events in your window. The interface that your filter function must support is described in "Application-Defined Functions" beginning on page 5-111.   |
| <code>procRefNum</code> | Contains a reference constant for use by your filter function.  |

**IMPORTANT**

Because the settings dialog box is a movable modal dialog box, you must provide an event filter function. ▲

**DESCRIPTION**

The `SGSettingsDialog` function instructs the sequence grabber to display its settings dialog box to the user. The sequence grabber works with one or more panel components to configure a specified channel component.

If the user clicks OK and the settings are acceptable to the panel and channel components, this function returns a result code of `noErr`. Because the user may change several channel configuration parameters, your application should retrieve new configuration information from the channel so that you can update any values you save, such as the channel's display boundaries or the channel device. In particular, the video rectangle for the channels may be adjusted.

**RESULT CODE**

`userCanceledErr`     -128     User canceled the dialog

**SEE ALSO**

You may retrieve or set the configuration of one or more channel components by using the `SGGetSettings` (described in the next section), `SGSetSettings` (described on page 5-50), `SGGetChannelSettings` (described on page 5-51), or `SGSetChannelSettings` function (described on page 5-52).

**SGGetSettings**

The `SGGetSettings` function retrieves the current settings of all channels used by the sequence grabber. The sequence grabber places all of this configuration information into a Movie Toolbox user data list.

```
pascal ComponentResult SGGetSettings (SeqGrabComponent s,
                                      UserData *ud, long flags);
```

- |       |   |
|-------|---|
| s     | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function.          |
| ud    | Contains a pointer. The sequence grabber returns a pointer to a Movie Toolbox user data list that contains the configuration information. Your application is responsible for disposing of this user data list when you are done with it. |
| flags | Reserved for Apple. Set this parameter to 0.  |

## Sequence Grabber Components

**DESCRIPTION**

The `SGGetSettings` function allows you to retrieve the sequence grabber's configuration information. The sequence grabber, in turn, retrieves configuration information for each of its channels and stores that information in a Movie Toolbox user data list. You may subsequently use the `SGSetSettings` function (described in the next section) to reconfigure the sequence grabber. You can store the settings (for example, in a Preferences file) by using the Movie Toolbox's `PutUserDataIntoHandle` function.

**RESULT CODES**

Memory Manager errors

**SEE ALSO**

You may retrieve the configuration of one channel component by using the `SGGetChannelSettings` function (described on page 5-51).

**SGSetSettings**

---

The `SGSetSettings` function allows you to configure a sequence grabber and its channels.

```
pascal ComponentResult SGSetSettings (SeqGrabComponent s,
                                     UserData ud, long flags);
```

|                    |  |
|--------------------|--|
| <code>s</code>     | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| <code>ud</code>    | Specifies a Movie Toolbox user data list that contains the configuration information to be used by the sequence grabber.   |
| <code>flags</code> | Reserved for Apple. Set this parameter to 0.   |

**DESCRIPTION**

The `SGSetSettings` function allows you to configure a sequence grabber. You provide this configuration information in a Movie Toolbox user data list. Typically, you obtain this configuration data from the `SGGetSettings` function, which is discussed in the previous section.

Note that the sequence grabber disposes of any of its current channels before applying this configuration information. It then opens connections to new channels as appropriate.

You can restore saved settings by using the Movie Toolbox's `NewUserDataFromHandle` function.



RESULT CODES

|                             |       |  |
|-----------------------------|-------|--|
| noDeviceForChannel          | -9400 | Channel component cannot find its device |
| couldntGetRequiredComponent | -9405 | Component not found                      |
| deviceCantMeetRequest       | -9408 | Device cannot support grabber            |

SEE ALSO

You may set the configuration of one channel component by using the `SGSetChannelSettings` function (described on page 5-52).

You may use the `SGGetIndChannel` function (described on page 5-33) to obtain information about each channel that the sequence grabber is using as a result of applying this new configuration.

SGGetChannelSettings

The `SGGetChannelSettings` function retrieves the current settings of one channel used by the sequence grabber. The sequence grabber places this configuration information into a Movie Toolbox user data list.

```
pascal ComponentResult SGGetChannelSettings (SeqGrabComponent s,
                                             SGChannel c,
                                             UserData *ud,
                                             long flags);
```

|       |   |
|-------|---|
| s     | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function.                                    |
| c     | Identifies the channel for this operation. You pass your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, discussed on page 5-31 and page 5-32, respectively. |
| ud    | Contains a pointer. The sequence grabber returns a pointer to a Movie Toolbox user data list that contains the configuration information.   |
| flags | Reserved for Apple. Set this parameter to 0.  |

DESCRIPTION

The `SGGetChannelSettings` function allows you to retrieve the configuration information for a single channel component. The channel component stores that information in a Movie Toolbox user data list. You may subsequently use the `SGSetChannelSettings` function to reconfigure the channel (this function is described next).

**RESULT CODES**

Memory Manager errors

**SEE ALSO**

You may retrieve the configuration of the entire sequence grabber, including all of its channels, by using the `SGGetSettings` function, described on page 5-49.

## SGSetChannelSettings

---

The `SGSetChannelSettings` function allows you to configure a sequence grabber channel.

```
pascal ComponentResult SGSetChannelSettings (SeqGrabComponent s,
                                             SGChannel c,
                                             UserData ud,
                                             long flags);
```

|                    |  |
|--------------------|--|
| <code>s</code>     | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function.                                     |
| <code>c</code>     | Identifies the channel to be configured. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, discussed on page 5-31 and page 5-32, respectively. |
| <code>ud</code>    | Specifies a Movie Toolbox user data list that contains the configuration information to be used by the channel component.  |
| <code>flags</code> | Reserved for Apple. Set this parameter to 0.   |

**DESCRIPTION**

The `SGSetChannelSettings` function allows you to configure a sequence grabber channel. You provide this configuration information in a Movie Toolbox user data list. Typically, you obtain this configuration data from the `SGGetChannelSettings` function, which is discussed in the previous section.

## Sequence Grabber Components

## RESULT CODES

|  |       |  |
|--|-------|--|
| <code>noDeviceForChannel</code>          | -9400 | Channel component cannot find its device |
| <code>couldntGetRequiredComponent</code> | -9405 | Component not found                      |
| <code>deviceCantMeetRequest</code>       | -9408 | Device cannot support grabber            |

## SEE ALSO

You may set the configuration of all of the sequence grabber's channels by using the `SGSetSettings` function. This function is described on page 5-50.

## Working With Sequence Grabber Characteristics

The characteristics that govern a sequence grabber operation fall into two main categories: those that apply to the sequence grabber component, and those that apply to an individual channel that has been created for the sequence grabber. Sequence grabber components provide a number of functions in each category. This section describes the functions that allow you to configure the characteristics of the sequence grabber component. See “Working With Channel Characteristics” beginning on page 5-58 for information about functions that apply to a single channel.

Use the `SGSetMaximumRecordTime` function to limit the duration of a record operation. You can retrieve this time limit by calling the `SGGetMaximumRecordTime` function.

The `SGSetFlags` function allows you to set control flags that govern an operation. Use the `SGGetFlags` function to retrieve those flags.

You can obtain information about the progress of a record operation by calling the `SGGetStorageSpaceRemaining` and `SGGetTimeRemaining` functions.

You can retrieve a reference to the time base used by a sequence grabber component by calling the `SGGetTimeBase` function.

## SGSetMaximumRecordTime

You can limit the duration of a record operation by calling the `SGSetMaximumRecordTime` function. You specify the time limit as an exact number of Macintosh system ticks (each is approximately a sixtieth of a second). The most efficient technique for monitoring this time limit is to examine the result code from the `SGIdle` function, which is described on page 5-39. When the time limit expires, the sequence grabber component sets that result code to `grabTimeComplete`.

```
pascal ComponentResult SGSetMaximumRecordTime (SeqGrabComponent s,
                                                unsigned long ticks);
```

## Sequence Grabber Components

|                    |  |
|--------------------|--|
| <code>s</code>     | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| <code>ticks</code> | Specifies the maximum duration for the record operation, in system ticks. Set this parameter to 0 to remove the time limit from the operation.   |

**DESCRIPTION**

By default, there is no time limit on a record operation. If you do not set a limit, a record operation will run until it exhausts the Operating System resources or you call the `SGStop` function (described on page 5-40). Memory and disk space are the two major limiting factors.

You must call the `SGSetMaximumRecordTime` function before you start the record operation.

**SGGetMaximumRecordTime**

---

The `SGGetMaximumRecordTime` function allows you to determine the time limit you have set for a record operation.

```
pascal ComponentResult SGGetMaximumRecordTime (SeqGrabComponent s,
                                                unsigned long *ticks);
```

|                    |  |
|--------------------|--|
| <code>s</code>     | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| <code>ticks</code> | Contains a pointer to a long integer that is to receive a value indicating the maximum duration for the record operation, in system ticks. A value of 0 indicates that there is no time limit.                                   |

**SEE ALSO**

You set this time limit by calling the `SGSetMaximumRecordTime` function, which is described in the previous section.

## SGGetStorageSpaceRemaining

The `SGGetStorageSpaceRemaining` function allows you to monitor the amount of space remaining for use during a record operation. You can use this function to monitor the space being used so that you can limit the amount of space consumed by an operation. Alternatively, you can use the information you receive from this function to update a status display for the user.

```
pascal ComponentResult SGGetStorageSpaceRemaining
                                (SeqGrabComponent s,
                                unsigned long *bytes);
```

- `s` Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's `OpenDefaultComponent` or `OpenComponent` function.
- `bytes` Contains a pointer to a long integer that is to receive a value indicating the amount of space remaining for the current record operation. If you are recording to memory, this value contains information about the amount of memory remaining. If you are recording to a movie file, this value contains information about the amount of storage space available on the device that holds the file.

DESCRIPTION

The `SGGetStorageSpaceRemaining` function returns information that is appropriate for the output conditions you establish with the `SGSetDataOutput` function, which is described on page 5-26. If you are recording to memory, this function returns information about the amount of memory remaining. If you are recording to a movie file, this function returns information about the amount of storage space available on the device that holds the file.

You can call this function only after you have started a record operation.

RESULT CODE

|                                      |                    |  |
|--------------------------------------|--------------------|--|
| <code>seqGrabInfoNotAvailable</code> | <code>-9407</code> | Sequence grabber does not have this information at this time |
|--------------------------------------|--------------------|--|

## SGGetTimeRemaining

---

The `SGGetTimeRemaining` function allows you to obtain an estimate of the amount of recording time that remains for the current record operation. The sequence grabber component estimates this value based on the amount of storage remaining and the speed with which the record operation is consuming that space. This estimate improves as the record process continues. If you have limited the record time by calling the `SGSetMaximumRecordTime` function (see page 5-53 for details), `SGGetTimeRemaining` does not return a value that is greater than the limit you have set.

```
pascal ComponentResult SGGetTimeRemaining (SeqGrabComponent s,
                                           long *ticksLeft);
```

|                        |  |
|------------------------|--|
| <code>s</code>         | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| <code>ticksLeft</code> | Contains a pointer to a long integer that is to receive a value indicating an estimate of the amount of time remaining for the current record operation. This value is expressed in system ticks.                                |

### DESCRIPTION

You can call the `SGGetTimeRemaining` function only after you have started a record operation.

### SPECIAL CONSIDERATIONS

This function may take a relatively long time to execute. You should not call it too frequently—once per second is reasonable.

### RESULT CODE

|                                      |       |   |
|--------------------------------------|-------|---|
| <code>seqGrabInfoNotAvailable</code> | -9407 | Sequence grabber cannot support request |
|--------------------------------------|-------|---|

## SGGetTimeBase

---

The `SGGetTimeBase` function allows you to retrieve a reference to the time base that is being used by a sequence grabber component.

```
pascal ComponentResult SGGetTimeBase (SeqGrabComponent s,
                                       TimeBase *tb);
```

Sequence Grabber Components

|    |  |
|----|--|
| s  | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| tb | Contains a pointer to a time base record that is to receive information about the sequence grabber's time base.  |

DESCRIPTION

You can examine the time base to monitor an operation or to schedule events based on time values. However, you should not change this time base in any way.

SGSetFlags

---

The `SGSetFlags` function allows you to pass control information about the current operation to the sequence grabber component.

```
pascal ComponentResult SGSetFlags (SeqGrabComponent s,
                                   long sgFlags);
```

|         |  |
|---------|--|
| s       | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| sgFlags | Contains flags for the current operation. The following flag is defined (set unused flags to 0):   |

`sgFlagControlledGrab`  
Informs the sequence grabber component that you are working with a frame-addressable device to perform a controlled record operation. The sequence grabber and its channel components optimize their operation for this situation. This flag allows the sequence grabber component to trade off speed and quality. Set this flag to 1 if you are performing a controlled grab using a frame-addressable source device.

SGGetFlags

---

You can retrieve a sequence grabber's control flags by calling the `SGGetFlags` function.

```
pascal ComponentResult SGGetFlags (SeqGrabComponent s,
                                   long *sgFlags) ;
```

|   |  |
|---|--|
| s | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
|---|--|

## Sequence Grabber Components

|                                   |  |
|-----------------------------------|--|
| <code>sgFlags</code>              | Contains a pointer to a long integer that is to receive the control flags for the current operation. The following flag is defined (unused flags are set to 0):  |
| <code>sgFlagControlledGrab</code> | <p>Informs the sequence grabber component that you are working with a frame-addressable device to perform a controlled record operation. The sequence grabber and its channel components optimize their operation for this situation. This flag allows the sequence grabber component to trade off speed and quality. This flag is set to 1 if you are performing a controlled grab using a frame-addressable source device.</p> |

## SEE ALSO

You set these flags by calling the `SGSetFlags` function, which is described in the previous section.

## Working With Channel Characteristics

Sequence grabber components use channel components to obtain digitized data from external media. After you create a channel for a sequence grabber component (by calling the `SGNewChannel` function, which is described on page 5-31), you must configure that channel before you start a preview or record operation. The sequence grabber component provides a number of functions that allow you to configure the characteristics of a channel component. Several of these functions work on any channel component. This section discusses these general channel configuration functions.

In addition, sequence grabber components provide functions that are specific to the channel type. Apple currently provides two types of channel components: video channel components and sound channel components. See “Working With Video Channels” beginning on page 5-77 for information about the sequence grabber configuration functions that work only with video channels. See “Working With Sound Channels” beginning on page 5-92 for information about the sequence grabber configuration functions that work only with sound channels.

Use the `SGSetChannelUsage` function to specify how a channel is to be used. You can restrict a channel to use during record or preview operations. In addition, this function allows you to specify whether a channel plays during a record operation. The `SGGetChannelUsage` function enables you to determine a channel’s usage.

The `SGGetChannelInfo` function allows you to determine whether a channel has a visual or an audio representation.

The `SGSetChannelPlayFlags` function allows you to influence the speed and quality with which the sequence grabber displays captured data. The `SGGetChannelPlayFlags` function lets you determine these flag settings.



The `SGSetChannelMaxFrames` function establishes a limit on the number of frames that the sequence grabber will capture from a channel. The `SGGetChannelMaxFrames` function allows you to determine that limit.

The `SGSetChannelBounds` function allows you to set the display boundary rectangle for a channel. Use the `SGGetChannelBounds` function to determine a channel's boundary rectangle.

The `SGSetChannelVolume` function allows you to control a channel's sound volume. Use the `SGGetChannelVolume` function to determine a channel's volume.

The `SGSetChannelRefCon` function allows you to set the value of a reference constant that is passed to your callback functions (see "Video Channel Callback Functions" beginning on page 5-99 for information about the callback functions that are supported by video channels).

Use the `SGGetChannelSampleDescription` function to retrieve a channel's sample description. The `SGGetChannelTimeScale` function allows you to obtain the channel's time scale.

You can modify or retrieve the channel's clipping region by calling the `SGSetChannelClip` or `SGGetChannelClip` function, respectively. You can work with a channel's transformation matrix by calling the `SGSetChannelMatrix` and `SGGetChannelMatrix` functions.

## SGSetChannelUsage

The `SGSetChannelUsage` function specifies how a channel is to be used by the sequence grabber component. The sequence grabber component does not use a channel until you specify how it is to be used. You can specify that a channel is to be used for recording or previewing, or both. In addition, you can control whether the data captured by a channel is displayed during the record or preview operation.

```
pascal ComponentResult SGSetChannelUsage (SGChannel c,
                                           long usage);
```

|                    |  |
|--------------------|--|
| <code>c</code>     | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.   |
| <code>usage</code> | Contains flags (defined by the <code>SeqGrabUsageEnum</code> data type) specifying how the channel is to be used. You may set more than one of these flags to 1. Set unused flags to 0. The following flags are defined: |

`seqGrabRecord`

Indicates that the channel is to be used during record operations. Set this flag to 1 to use a channel for recording.

`seqGrabPreview`

Indicates that the channel is to be used during preview operations. Set this flag to 1 to use a channel for previewing.

Sequence Grabber Components

`seqGrabPlayDuringRecord`

Indicates that the sequence grabber may play the data captured by this channel during a record operation. If you set this flag to 1, the data from the channel may be played during the record operation, if the destination buffer is onscreen. Video data is displayed; sound data is played through the computer's speaker. However, playing the data may affect the quality of the recorded sequence by causing frames to be dropped. Set this flag to 0 to prevent the channel's data from being played during a record operation.

**DESCRIPTION**

You cannot call the `SGSetChannelUsage` function during a record or preview operation.

**RESULT CODES**

|                                       |       |  |
|---------------------------------------|-------|--|
| <code>notEnoughMemoryToGrab</code>    | -9403 | Insufficient memory for record operation     |
| <code>notEnoughDiskSpaceToGrab</code> | -9404 | Insufficient disk space for record operation |
| <code>deviceCantMeetRequest</code>    | -9408 | Device cannot support grabber                |

## SGGetChannelUsage

---

The `SGGetChannelUsage` function allows you to determine how a channel is to be used by the sequence grabber component.

```
pascal ComponentResult SGGetChannelUsage (SGChannel c,
                                           long *usage);
```

|                                      |   |                            |  |                             |   |                                      |  |
|--------------------------------------|---|----------------------------|--|-----------------------------|---|--------------------------------------|--|
| <code>c</code>                       | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.  |                            |  |                             |   |                                      |  |
| <code>usage</code>                   | Contains a pointer to flags indicating how the channel is to be used. More than one flag may be set to 1; unused flags are set to 0. The following flags are defined: <table> <tr> <td><code>seqGrabRecord</code></td><td>Indicates that the channel is used during record operations.</td></tr> <tr> <td><code>seqGrabPreview</code></td><td>Indicates that the channel is used during preview operations.</td></tr> <tr> <td><code>seqGrabPlayDuringRecord</code></td><td>Indicates that the sequence grabber component plays the data captured by this channel during a record operation.</td></tr> </table> | <code>seqGrabRecord</code> | Indicates that the channel is used during record operations. | <code>seqGrabPreview</code> | Indicates that the channel is used during preview operations. | <code>seqGrabPlayDuringRecord</code> | Indicates that the sequence grabber component plays the data captured by this channel during a record operation. |
| <code>seqGrabRecord</code>           | Indicates that the channel is used during record operations.  |                            |  |                             |   |                                      |  |
| <code>seqGrabPreview</code>          | Indicates that the channel is used during preview operations.   |                            |  |                             |   |                                      |  |
| <code>seqGrabPlayDuringRecord</code> | Indicates that the sequence grabber component plays the data captured by this channel during a record operation.  |                            |  |                             |   |                                      |  |

**SEE ALSO**

You establish a channel's usage by calling the `SGSetChannelUsage` function, described in the previous section.

**SGGetChannelInfo**

---

The `SGGetChannelInfo` function allows you to determine how a channel's data is represented to the user—as visual or audio data, or both.

```
pascal ComponentResult SGGetChannelInfo (SGChannel c,
                                         long *channelInfo);
```

**c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

**channelInfo** Contains a pointer to a long integer that is to receive channel information flags. More than one flag may be set to 1. Unused flags are set to 0. The following flags are defined:

`seqGrabHasBounds`

Indicates that the channel has a visual representation. If this flag is set to 1, the channel has a visual representation.

`seqGrabHasVolume`

Indicates that the channel has an audio representation. If this flag is set to 1, the channel has an audio representation.

`seqGrabHasDiscreteSamples`

Indicates that the channel data is organized into discrete frames. If this flag is set to 1, you can use the `SGSetChannelMaxFrames` function (see page 5-63) to limit the number of frames processed in a record operation or the rate at which those frames are processed. If this flag is set to 0, the channel data is not organized into frames. Therefore, you can only limit a record operation by setting the maximum time for the operation.

**SGSetChannelPlayFlags**

---

The `SGSetChannelPlayFlags` function allows you to influence the speed and quality with which the sequence grabber displays data from a channel.

```
pascal ComponentResult SGSetChannelPlayFlags (SGChannel c,
                                              long playFlags);
```

## Sequence Grabber Components

|                        |   |
|------------------------|---|
| <code>c</code>         | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.  |
| <code>playFlags</code> | Specifies a long integer that contains flags that influence channel playback. The following values are defined—you must use one of these values: <ul style="list-style-type: none"> <li><code>channelPlayNormal</code><br/>Instructs the channel component to use its default playback methodology.</li> <li><code>channelPlayFast</code><br/>Instructs the channel component to sacrifice playback quality in order to achieve the specified playback rate.</li> <li><code>channelPlayHighQuality</code><br/>Instructs the channel component to play the channel's data at the highest possible quality—this option sacrifices playback rate for the sake of image quality. This option may reduce the amount of processor time available for recording. This option does not affect the quality of the recorded data, however.</li> </ul> <p>The following flag is defined—you may use this flag with any of the values defined for this parameter (set unused flags to 0):</p> <ul style="list-style-type: none"> <li><code>channelPlayAllData</code><br/>Instructs the channel component to try to play all of the data it captures, even the data that is stored in offscreen buffers. This option is useful when you want to be sure that the user sees as much of the captured data as possible. Set this flag to 1 to play all the captured data. You may combine this flag with any of the values defined for the <code>playFlags</code> parameter.</li> </ul> |

**DESCRIPTION**

The `SGSetChannelPlayFlags` function does not affect the quality of a record operation.

**SPECIAL CONSIDERATIONS**

You cannot call this function during a record operation; you can call it during a preview operation.

## SGGetChannelPlayFlags

---

The `SGGetChannelPlayFlags` function allows you to retrieve the playback control flags that you set with the `SGSetChannelPlayFlags` function, which is described in the previous section.

```
pascal ComponentResult SGGetChannelPlayFlags (SGChannel c,
                                              long *playFlags);
```

**c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

**playFlags** Contains a pointer to a long integer that is to receive flags that influence channel playback. The following values are defined:

`channelPlayNormal`

The channel component uses its default playback methodology.

`channelPlayFast`

The channel component sacrifices playback quality in order to achieve the specified playback rate.

`channelPlayHighQuality`

The channel component plays the channel's data at the highest possible quality—this option sacrifices playback rate for the sake of image quality. This option may reduce the amount of processor time available for recording. This option does not affect the quality of the recorded data, however.

The following flag is defined and may be used with any of the values defined for this parameter (unused flags are set to 0):

`channelPlayAllData`

The channel component tries to play all of the data it captures, even the data that is stored in offscreen buffers. This option is useful when you want to be sure that the user sees as much of the captured data as possible.

## SGSetChannelMaxFrames

---

The `SGSetChannelMaxFrames` function allows you to limit the number of frames that the sequence grabber will capture from a specified channel. This function works only with channels that have data that is organized into frames, such as video data from a video disc.

```
pascal ComponentResult SGSetChannelMaxFrames (SGChannel c,
                                              long frameCount);
```

## Sequence Grabber Components

|                         |  |
|-------------------------|--|
| <code>c</code>          | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31. |
| <code>frameCount</code> | Specifies the maximum number of frames to capture during the preview or record operation. Set this value to <code>-1</code> to remove the limit.                       |

## DESCRIPTION

You can use the `SGSetChannelMaxFrames` function in the context of a time-based function to control the number of frames you collect for each unit of time. For example, if you want to collect one frame of data per second, you can create a function that executes once per second. That function should call `SGSetChannelMaxFrames` to set the maximum frame count to 1. Your application can determine when the frame is captured by calling the `SGGetChannelMaxFrames` function and detecting when that function returns a value of 0. The `SGGetChannelMaxFrames` function is described in the next section.

You may use this function only after you have prepared the sequence grabber component for a record operation or during an active record operation. Note that sequence grabber components clear this value when you prepare for a record operation.

## SEE ALSO

You can determine whether a channel's data is organized into frames by calling the `SGGetChannelInfo` function, which is described on page 5-61.

## RESULT CODES

|                                      |                    |                                 |
|--------------------------------------|--------------------|---------------------------------|
| <code>paramErr</code>                | <code>-50</code>   | Invalid parameter specified     |
| <code>cantDoThatInCurrentMode</code> | <code>-9402</code> | Request invalid in current mode |

**SGGetChannelMaxFrames**

---

The `SGGetChannelMaxFrames` function allows you to determine the number of frames left to be captured from a specified channel.

```
pascal ComponentResult SGGetChannelMaxFrames (SGChannel c,
                                              long *frameCount);
```

|                |  |
|----------------|--|
| <code>c</code> | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31. |
|----------------|--|

## Sequence Grabber Components

frameCount

Contains a pointer to a long integer that is to receive a value specifying the number of frames left to be captured during the preview or record operation. If the returned value is -1, the sequence grabber captures as many frames as it can.

## SEE ALSO

You set the starting value by calling the `SGSetChannelMaxFrames` function, which is described in the previous section.

## RESULT CODE

|                                      |       |   |
|--------------------------------------|-------|---|
| <code>seqGrabInfoNotAvailable</code> | -9407 | Sequence grabber component cannot support request |
|--------------------------------------|-------|---|

## SGSetChannelBounds

---

The `SGSetChannelBounds` function allows you to specify a channel's display boundary rectangle. This rectangle defines the destination for data from this channel. This rectangle is defined in the graphics world you establish by calling the `SGSetGWorld` function, described on page 5-29.

```
pascal ComponentResult SGSetChannelBounds (SGChannel c,
                                           const Rect *bounds);
```

|                     |   |
|---------------------|---|
| <code>c</code>      | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.  |
| <code>bounds</code> | Contains a pointer to a rectangle that defines the channel's display boundary rectangle. This rectangle is defined in the graphics world you establish when you call the <code>SGSetGWorld</code> function, described on page 5-29. |

## DESCRIPTION

You cannot call the `SGSetChannelBounds` function during a record operation.

## SPECIAL CONSIDERATIONS

The `SGSetChannelBounds` function adjusts the channel matrix, as appropriate.

## Sequence Grabber Components

## RESULT CODES

|                                      |       |  |
|--------------------------------------|-------|--|
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode          |
| <code>notEnoughMemoryToGrab</code>   | -9403 | Insufficient memory for record operation |
| <code>deviceCantMeetRequest</code>   | -9408 | Device cannot support grabber component  |

**SGGetChannelBounds**

---

The `SGGetChannelBounds` function allows you to determine a channel's display boundary rectangle.

```
pascal ComponentResult SGGetChannelBounds (SGChannel c,
                                           Rect *bounds);
```

|                     |  |
|---------------------|--|
| <code>c</code>      | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.   |
| <code>bounds</code> | Contains a pointer to a rectangle structure that is to receive information about the channel's display boundary rectangle. This rectangle is defined in the graphics world that you establish when you call the <code>SGSetGWorld</code> function. |

## DESCRIPTION

You set the boundary rectangle by calling the `SGSetChannelBounds` function, which is described in the previous section. This rectangle is defined in the graphics world that you establish by calling the `SGSetGWorld` function, described on page 5-29.

**SGSetChannelVolume**

---

The `SGSetChannelVolume` function sets a channel's sound volume.

```
pascal ComponentResult SGSetChannelVolume (SGChannel c,
                                           short volume);
```

|                     |  |
|---------------------|--|
| <code>c</code>      | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.   |
| <code>volume</code> | Specifies the volume setting of the channel represented as a 16-bit, fixed-point number. The high-order 8 bits contain the integer part of the value; the low-order 8 bits contain the fractional part. Volume values range from -1.0 to 1.0. Negative values play no sound but preserve the absolute value of the volume setting. |



**DESCRIPTION**

The sequence grabber component uses this volume setting during playback—this setting does not affect the record level or the volume of the track in the recorded QuickTime movie.

**SGGetChannelVolume**

---

The `SGGetChannelVolume` function allows you to determine a channel's sound volume setting.

```
pascal ComponentResult SGGetChannelVolume (SGChannel c,
                                           short *volume);
```

- |                     |  |
|---------------------|--|
| <code>c</code>      | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.   |
| <code>volume</code> | Contains a pointer to an integer that is to receive the volume setting of the channel represented as a 16-bit, fixed-point number. The high-order 8 bits contain the integer part of the value; the low-order 8 bits contain the fractional part. Volume values range from -1.0 to 1.0. Negative values play no sound but preserve the absolute value of the volume setting. |

**SEE ALSO**

You establish the volume setting by calling the `SGSetChannelVolume` function, described in the previous section.

**SGSetChannelRefCon**

---

The `SGSetChannelRefCon` function allows you to set the value of a reference constant that is passed to your callback functions (see “Video Channel Callback Functions” beginning on page 5-99 for information about the callback functions that are supported by video channels).

```
pascal ComponentResult SGSetChannelRefCon (SGChannel c,
                                           long refCon);
```

- |                |  |
|----------------|--|
| <code>c</code> | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31. |
|----------------|--|

## Sequence Grabber Components

**refCon** Specifies a reference constant value that is to be passed to your callback functions for this channel. See “Video Channel Callback Functions” on page 5-99 for information about the callback functions that are supported by video channels. Sound channels do not support callback functions.

**SPECIAL CONSIDERATIONS**

Sound channels do not support callback functions.

**SGGetChannelSampleDescription**

---

The `SGGetChannelSampleDescription` function allows you to retrieve a channel's sample description.

```
pascal ComponentResult SGGetChannelSampleDescription
                        (SGChannel c, Handle sampleDesc);
```

**c** Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the `SGNewChannel` or `SGNewChannelFromComponent` function, described on page 5-31 and page 5-32, respectively.

**sampleDesc** Specifies a handle that is to receive the sample description.

**DESCRIPTION**

The `SGGetChannelSampleDescription` function allows you to retrieve a channel's current sample description. You may call this function only when the channel is prepared to record or is actually recording.

The channel returns a sample description that is appropriate to the type of data being captured. For video channels, the channel component returns an Image Compression Manager image description structure; for sound channels, you receive a sound description structure, as defined by the Movie Toolbox.

**RESULT CODE**

|                                      |                    |                                 |
|--------------------------------------|--------------------|---------------------------------|
| <code>cantDoThatInCurrentMode</code> | <code>-9402</code> | Request invalid in current mode |
|--------------------------------------|--------------------|---------------------------------|

**SGGetChannelTimeScale**

---

The `SGGetChannelTimeScale` function allows you to retrieve a channel's time scale.

```
pascal ComponentResult SGGetChannelTimeScale (SGChannel c,
                                              TimeScale *scale);
```

|                    |  |
|--------------------|--|
| <code>c</code>     | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function; these functions are described on page 5-31 and page 5-32, respectively. |
| <code>scale</code> | Contains a pointer to a time scale structure. The channel component places information about its time scale into this structure.   |

**DESCRIPTION**

The time scale you obtain by calling the `SGGetChannelTimeScale` typically corresponds to the time scale of the media that has been created by the channel. You can use this time scale in your data function, which you assign with the `SGSetDataProc` function (discussed on page 5-35).

## SGSetChannelClip

---

The `SGSetChannelClip` function allows you to set a channel's clipping region.

```
pascal ComponentResult SGSetChannelClip (SGChannel c,
                                         RgnHandle theClip);
```

|                      |   |
|----------------------|---|
| <code>c</code>       | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, described on page 5-31 and page 5-32, respectively.          |
| <code>theClip</code> | Contains a handle to the new clipping region. Set this parameter to <code>nil</code> to remove the current clipping region. The channel component makes a copy of this handle; it is your application's responsibility to dispose of this handle when you are finished with it. |

**DESCRIPTION**

The `SGSetChannelClip` function allows you to apply a clipping region to a channel's display region. By default, channel components do not apply a clipping region to their displayed image.

**SEE ALSO**

You may retrieve a channel's clipping region by calling the `SGGetChannelClip` function, described in the next section.

## SGGetChannelClip

---

The `SGGetChannelClip` function allows you to retrieve a channel's clipping region.

```
pascal ComponentResult SGGetChannelClip (SGChannel c,
                                         RgnHandle *theClip);
```

- |                      |  |
|----------------------|--|
| <code>c</code>       | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, described on page 5-31 and page 5-32, respectively. |
| <code>theClip</code> | Contains a pointer to a handle that is to receive the clipping region. Your application is responsible for disposing of this handle. If there is no clipping region, the channel component sets this handle to <code>nil</code> .                                      |

**Note**

Some devices may not support clipping. ♦

**SEE ALSO**

You may set a channel's clipping region by calling the `SGSetChannelClip` function, which is discussed in the previous section.

## SGSetChannelMatrix

---

The `SGSetChannelMatrix` function allows you to set a channel's display transformation matrix.

```
pascal ComponentResult SGSetChannelMatrix (SGChannel c,
                                           const MatrixRecord *m);
```

- |                |  |
|----------------|--|
| <code>c</code> | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, discussed on page 5-31 and page 5-32, respectively. |
| <code>m</code> | Contains a pointer to a matrix structure, as defined by the Movie Toolbox (see the chapter "Movie Toolbox" in <i>Inside Macintosh: QuickTime</i> for more information about matrix structures). Set this parameter to <code>nil</code> to select the identity matrix.  |

**DESCRIPTION**

The `SGSetChannelMatrix` function allows you to specify a display transformation matrix for a video channel. The channel uses this matrix to transform its video image into the destination window. If the channel cannot accommodate your matrix, it returns an appropriate result code. Note that you may not call this function when you are recording.

Other channel component functions may affect this matrix. The `SGSetChannelBounds` function sets the matrix values so that the matrix maps the channel's output to the channel's boundary rectangle (this function is discussed beginning on page 5-65). The `SGSetVideoRect` function modifies the matrix so that the specified video rectangle appears in the existing destination rectangle (see page 5-78 for more information about this function).

**RESULT CODES**

|                                    |       |                               |
|------------------------------------|-------|-------------------------------|
| <code>matrixErr</code>             | -2203 | Invalid matrix                |
| <code>deviceCantMeetRequest</code> | -9408 | Device cannot support grabber |

**SEE ALSO**

You may retrieve a channel's matrix by calling the `SGGetChannelMatrix` function, which is discussed next.

**SGGetChannelMatrix**

The `SGGetChannelMatrix` function allows you to retrieve a channel's display transformation matrix.

```
pascal ComponentResult SGGetChannelMatrix (SGChannel c,
                                           MatrixRecord *m);
```

|                |  |
|----------------|--|
| <code>c</code> | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, described on page 5-31 and page 5-32, respectively. |
| <code>m</code> | Contains a pointer to a matrix structure, as defined by the Movie Toolbox (see "Movie Toolbox" in <i>Inside Macintosh: QuickTime</i> for more information about matrix structures). The channel component places its current matrix values into this matrix structure. |

**SEE ALSO**

You may set a channel's matrix by calling the `SGSetChannelMatrix` function, which is discussed in the previous section.

## Working With Channel Devices

---

Sequence grabbers provide a number of functions that allow you to determine the device that is attached to a given sequence grabber channel. These devices allow the channel component to control the digitizing equipment. For example, video channels use video digitizer components, and sound channels use sound input drivers. Your application can use these routines to present a list of available devices to the user, allowing the user to select a specific device for each channel.

You may use the `SGGetChannelDeviceList` function to retrieve a list of devices that may be used with a specified channel. You dispose of this device list by calling the `SGDisposeDeviceList` function. You can place one or more device names into a menu by calling the `SGAppendDeviceListToMenu` function. You can use the `SGSetChannelDevice` function to assign a device to a channel.

Some of these functions use a device list structure to pass information about one or more channel devices. The `SGDeviceListRecord` data type defines the format of the device list structure.

```
typedef struct SGDeviceListRecord {
    short          count;           /* count of devices */
    short          selectedIndex;   /* current device */
    long           reserved;        /* set to 0 */
    SGDeviceName   entry[1];        /* device names */
} SGDeviceListRecord, *SGDeviceListPtr, **SGDeviceList;
```

### Field descriptions

|                            |   |
|----------------------------|---|
| <code>count</code>         | Indicates the number of devices described by this structure. The value of this field corresponds to the number of entries in the device name array defined by the <code>entry</code> field.   |
| <code>selectedIndex</code> | Identifies the currently active device. The value of this field corresponds to the appropriate entry in the device name array defined by the <code>entry</code> field. Note that this value is 0-relative; that is, the first entry has an index number of 0, the second's value is 1, and so on. |
| <code>reserved</code>      | Reserved for Apple. Always set to 0.  |
| <code>entry</code>         | Contains an array of device name structures. Each structure corresponds to one valid device. The <code>count</code> field indicates the number of entries in this array. The <code>SGDeviceName</code> data type defines the format of a device name structure; this data type is discussed next. |

Device list structures contain an array of device name structures. Each device name structure identifies a single device that may be used by the channel. The `SGDeviceName` data type defines the format of a device name structure.

Sequence Grabber Components

```
typedef struct SGDeviceName {
    Str63      name;           /* device name */
    Handle     icon;           /* device icon */
    long       flags;          /* flags */
    long       refCon;         /* set to 0 */
    long       reserved;       /* set to 0 */
} SGDeviceName;
```

Field descriptions

|          |  |
|----------|--|
| name     | Contains the name of the device. For video digitizer components, this field contains the component's name as specified in the component resource. For sound input drivers, this field contains the driver name.  |
| icon     | Contains a handle to the device's icon. Some devices may support an icon, which you may choose to present to the user. If the device does not support an icon, or if you choose not to retrieve this information (by setting the <code>sgDeviceListWithIcons</code> flag to 0 when you call the <code>SGGetChannelDeviceList</code> function), this field is set to nil. |
| flags    | Reflects the current status of the device. The sequence grabber sets these flags when you retrieve a device list. The following flag is defined:<br><div>sgDeviceNameFlagDeviceUnavailable<br/>When set to 1, this flag indicates that this device is not currently available.</div>   |
| refCon   | Reserved for Apple. Always set to 0.   |
| reserved | Reserved for Apple. Always set to 0.   |

SGGetChannelDeviceList

The `SGGetChannelDeviceList` function allows you to retrieve a list of the devices that are valid for a specified channel.

```
pascal ComponentResult SGGetChannelDeviceList (SGChannel c,
                                                long selectionFlags,
                                                SGDeviceList *list);
```

|   |  |
|---|--|
| c | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, discussed on page 5-31 and page 5-32, respectively. |
|---|--|

## Sequence Grabber Components

`selectionFlags`

Controls the data returned for each device. The following flags are defined:

`sgDeviceListWithIcons`

Specifies whether you want to retrieve an icon for each device. If you set this flag to 1, the sequence grabber returns an icon for each device in the list, in the `icon` field. If you set this flag to 0, the sequence grabber sets the `icon` fields to 0.

`sgDeviceListDontCheckAvailability`

Controls whether the sequence grabber verifies that each device is currently available. If you set this flag to 1, the sequence grabber does not check the availability of each device. Otherwise, the sequence grabber checks each device's availability, and sets the `sgDeviceNameFlagDeviceUnavailable` flag appropriately in each device name structure that is returned.

Note that checking device availability slows this function. In general, however, you should check availability if you plan to present a list of devices to the user. Otherwise, the user may select a device that is unavailable.

`list`

Defines a pointer to a device list structure pointer. The sequence grabber creates a device name structure and returns a pointer to that structure in the field referred to by this parameter. When you are done with the list, use the `SGDisposeDeviceList` function (described in the next section) to dispose of the memory used by the list.

**DESCRIPTION**

This function allows you to retrieve a list of the devices that may be used with a channel. Each entry in this list identifies a valid device by name. Your application may then place these device names into a menu using the `SGAppendDeviceListToMenu` function, which is described on page 5-75.

This function can be useful for retrieving the name of the current device. Retrieve the device list and use the `selectedIndex` field to determine which device is currently in use.

**RESULT CODES**

Memory Manager errors

**SEE ALSO**

When you are done with the list, use the `SGDisposeDeviceList` function to dispose of the memory used by the list. This function is discussed next.



## SGDisposeDeviceList

---

The `SGDisposeDeviceList` function allows you to dispose of a device list.

```
pascal ComponentResult SGDisposeDeviceList (SeqGrabComponent s,  
                                             SGDeviceList list);
```

- |                   |  |
|-------------------|--|
| <code>s</code>    | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function. |
| <code>list</code> | Defines a pointer to a device list structure pointer. The sequence grabber disposes of the memory used by the device list structure.   |

### DESCRIPTION

You must use this function to dispose of the memory used by a device list structure. Do not use Memory Manager functions to do so.

### RESULT CODES

Memory Manager errors

## SGAppendDeviceListToMenu

---

The `SGAppendDeviceListToMenu` function allows you to place a list of device names into a specified menu.

```
pascal ComponentResult SGAppendDeviceListToMenu  
                        (SeqGrabComponent s,  
                        SGDeviceList list, MenuHandle mh);
```

- |                   |   |
|-------------------|---|
| <code>s</code>    | Specifies the component instance that identifies your connection to the sequence grabber component. You obtain this value from the Component Manager's <code>OpenDefaultComponent</code> or <code>OpenComponent</code> function.  |
| <code>list</code> | Defines a pointer to a device list structure pointer. The sequence grabber appends the name of each device in the list to the menu specified by the <code>mh</code> parameter. If the <code>sgDeviceNameFlagDeviceUnavailable</code> flag is set to 1 for a device in the list, the sequence grabber disables the menu item corresponding to that device. |
| <code>mh</code>   | Specifies the menu to which the device names are to be appended.  |

## Sequence Grabber Components

**DESCRIPTION**

You may use the `SGAppendDeviceListToMenu` function to present a list of valid devices to the user. The user may then select a device from the list. You can assign that device to a channel by calling the `SGSetChannelDevice` function. Note that, if you choose to have the sequence grabber check the availability of each device (by setting the `sgDeviceListDontCheckAvailability` flag to 0 with the `SGGetChannelDeviceList` function), the sequence grabber will disable menu items that correspond to unavailable devices. This prevents the user from selecting a device that cannot be used.

**RESULT CODE**

`paramErr`     -50     Invalid parameter value

**SEE ALSO**

You obtain the device list by calling the `SGGetChannelDeviceList` function, which is discussed on page 5-73.

**SGSetChannelDevice**

---

The `SGSetChannelDevice` function allows you to assign a device to a channel.

```
pascal ComponentResult SGSetChannelDevice (SGChannel c,
                                           StringPtr name);
```

|                   |  |
|-------------------|--|
| <code>c</code>    | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, discussed on page 5-31 and page 5-32, respectively. |
| <code>name</code> | Points to the device's name string. This name is contained in the <code>name</code> field of the appropriate device name structure in the device list.   |

**DESCRIPTION**

When you call the `SGSetChannelDevice` function, the sequence grabber channel tries to use the specified device, in place of the device currently in use. You must obtain the device name from the channel's device list.

**RESULT CODES**

|                                    |       |                               |
|------------------------------------|-------|-------------------------------|
| <code>paramErr</code>              | -50   | Invalid parameter value       |
| <code>deviceCantMeetRequest</code> | -9408 | Device cannot support grabber |

**SEE ALSO**

You obtain the device list by calling the `SGGetChannelDeviceList` function, which is described on page 5-73.

## Working With Video Channels

---

Sequence grabber components provide a number of functions that allow you to configure the grabber's video channels. This section describes these configuration functions, which you can use only with video channels. You can determine whether a channel has a visual representation by calling the `SGGetChannelInfo` function, which is described on page 5-61. If you want to configure a sound channel, use the functions described in "Working With Sound Channels" beginning on page 5-92. If you want to configure general attributes of a channel, use the functions described in "Working With Channel Characteristics" beginning on page 5-58.

The `SGGetSrcVideoBounds` function allows you to determine the coordinates of the source video boundary rectangle. This rectangle defines the size of the source video image being captured by the video channel. You can use the `SGSetVideoRect` function to specify a part of the source video boundary rectangle to be captured by the channel. The `SGGetVideoRect` function allows you to determine the active source video rectangle.

Typically, the sequence grabber component uses the Image Compression Manager to compress the video data it captures. You can control many aspects of this image-compression process. Use the `SGSetVideoCompressorType` function to specify the type of image compressor to use. You can determine the type of image compressor currently in use by calling the `SGGetVideoCompressorType` function. You can specify a particular image compressor and set many image-compression parameters by calling the `SGSetVideoCompressor` function. You can determine which image compressor is being used and its parameter settings by calling the `SGGetVideoCompressor` function.

The channel components that supply video data to a sequence grabber component typically work with a video digitizer component (see the chapter "Video Digitizer Components" in this book for a complete description of video digitizer components). Sequence grabber components provide functions that allow you to work with a channel's video digitizer component. You can use the `SGGetVideoDigitizerComponent` function to determine which video digitizer component is supplying data to a specified channel component. You can set a channel's video digitizer by calling the `SGSetVideoDigitizerComponent` function. If you change any video digitizer settings by calling the video digitizer component directly, you should inform the sequence grabber component by calling the `SGVideoDigitizerChanged` function.

## Sequence Grabber Components

Some video source data may contain unacceptable levels of visual noise or artifacts. One technique for removing this noise is to capture the image and then reduce it in size. During the size reduction process, the noise can be filtered out. Sequence grabber components provide functions that allow you to filter the input video data.

The `SGSetCompressBuffer` function sets a filter buffer for a video channel. The `SGGetCompressBuffer` function returns information about your filter buffer.

You can work with a video channel's frame rate by calling the `SGSetFrameRate` and `SGGetFrameRate` functions. You can control whether a channel uses an offscreen buffer by calling the `SGSetUseScreenBuffer` and `SGGetUseScreenBuffer` functions.

## SGGetSrcVideoBounds

---

The `SGGetSrcVideoBounds` function allows you to determine the size of the source video boundary rectangle. This rectangle defines the size of the source video image. For video channel components that work with video digitizer components, this rectangle corresponds to the video digitizer's active source rectangle (see the chapter "Video Digitizer Components" in this book for more information).

```
pascal ComponentResult SGGetSrcVideoBounds (SGChannel c, Rect *r);
```

|                |  |
|----------------|--|
| <code>c</code> | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31. |
| <code>r</code> | Contains a pointer to a rectangle structure that is to receive information about the source video boundary rectangle.  |

### RESULT CODE

|                       |                  |                             |
|-----------------------|------------------|-----------------------------|
| <code>paramErr</code> | <code>-50</code> | Invalid parameter specified |
|-----------------------|------------------|-----------------------------|

## SGSetVideoRect

---

The `SGSetVideoRect` function allows you to specify a part of the source video image that is to be captured by the sequence grabber component. This rectangle must reside within the boundaries of the source video boundary rectangle. You obtain the dimensions of the source video boundary rectangle by calling the `SGGetSrcVideoBounds` function, described in the previous section. If you do not use this function to set a source rectangle, the sequence grabber component captures the entire video image, as defined by the source video boundary rectangle.

```
pascal ComponentResult SGSetVideoRect (SGChannel c, Rect *r);
```

- c Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.
- r Contains a pointer to the dimensions of the rectangle that defines the portion of the source video image to be captured. This rectangle must lie within the boundaries of the source video boundary rectangle, which you can obtain by calling the `SGGetSrcVideoBounds` function.

DESCRIPTION

For video channel components that receive their data from video digitizer components, this function sets the video digitizer component's digitizer rectangle. See the chapter "Video Digitizer Components" in this book for information about video digitizer components.

You cannot call this function during a record operation.

RESULT CODES

|                                      |       |  |
|--------------------------------------|-------|--|
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode          |
| <code>notEnoughMemoryToGrab</code>   | -9403 | Insufficient memory for record operation |

SGGetVideoRect

The `SGGetVideoRect` function allows you to determine the portion of the source video image that is to be captured. Use the `SGSetVideoRect` function, which is described in the previous section, to set the dimensions of this rectangle.

```
pascal ComponentResult SGGetVideoRect (SGChannel c, Rect *r);
```

- c Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.
- r Contains a pointer to a rectangle structure that is to receive the dimensions of the rectangle that defines the portion of the source video image to be captured.

DESCRIPTION

If you have not set a source rectangle, the sequence grabber captures the entire source video image, as defined by the source video boundary rectangle.

SEE ALSO

You can obtain the dimensions of the source video boundary rectangle by calling the `SGGetSrcVideoBounds` function, described on page 5-78.

## SGSetVideoCompressorType

---

The `SGSetVideoCompressorType` function allows you to specify the type of image compression to be applied to the captured video images.

```
pascal ComponentResult SGSetVideoCompressorType (SGChannel c,
                                                  OSType compressorType);
```

**c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

**compressorType** Specifies the type of image compression to use. The value of this parameter must correspond to one of the image compressor types supported by the Image Compression Manager. Currently, six `CodecType` values are provided by Apple. You should use the `GetCodecNameList` function to retrieve these names, so that your application can take advantage of new compressor types that may be added in the future. For each `CodecType` value in the following list, the corresponding compression method is also identified by its text string name.

| Compressor type | Compressor name          |
|-----------------|--------------------------|
| 'rpza'          | video compressor         |
| 'jpeg'          | photo compressor         |
| 'rle '          | animation compressor     |
| 'raw '          | raw compressor           |
| 'smc '          | graphics compressor      |
| 'cvid'          | compact video compressor |

See the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime* for information about valid compressor types. If this value is set to 0, the default compression type is selected.

### DESCRIPTION

In addition, the `SGSetVideoCompressorType` function resets all image-compression parameters to their default values. You can then use the `SGSetVideoCompressor` function, described on page 5-82, to change the compression parameters.

### SPECIAL CONSIDERATIONS

You cannot call the `SGSetVideoCompressorType` function during a record operation or after you have prepared the sequence grabber component for a record operation (by calling the `SGPrepare` function, described on page 5-43).

**RESULT CODES**

|                                      |       |  |
|--------------------------------------|-------|--|
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode          |
| <code>notEnoughMemoryToGrab</code>   | -9403 | Insufficient memory for record operation |
| <code>deviceCantMeetRequest</code>   | -9408 | Device cannot support grabber            |

**SGGetVideoCompressorType**

---

The `SGGetVideoCompressorType` function allows you to determine the type of image compression that is being applied to a channel's video data.

```
pascal ComponentResult SGGetVideoCompressorType (SGChannel c,
                                                  OSType *compressorType);
```

**c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

**compressorType** Contains a pointer to an `OSType` field that is to receive information about the type of image compression to use. The returned value must correspond to one of the image compressor types supported by the Image Compression Manager. Currently, six `CodecType` values are provided by Apple. You should use the `GetCodecNameList` function to retrieve these names, so that your application can take advantage of new compressor types that may be added in the future. For each `CodecType` value in the following list, the corresponding compression method is also identified by its text string name.

| Compressor type | Compressor name          |
|-----------------|--------------------------|
| 'rpza'          | video compressor         |
| 'jpeg'          | photo compressor         |
| 'rle '          | animation compressor     |
| 'raw '          | raw compressor           |
| 'smc '          | graphics compressor      |
| 'cvid'          | compact video compressor |

See the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime* for information about valid compressor types.

**SEE ALSO**

You can set the image-compression type by calling the `SGSetVideoCompressorType` function, which is described in the previous section.

## SGSetVideoCompressor

---

The `SGSetVideoCompressor` function allows you to specify many of the parameters that control image compression of the video data captured by a video channel.

```
pascal ComponentResult SGSetVideoCompressor (SGChannel c,
                                             short depth,
                                             CompressorComponent compressor,
                                             CodecQ spatialQuality,
                                             CodecQ temporalQuality,
                                             long keyFrameRate);
```

- `c` Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.
- `depth` Specifies the depth at which the image is likely to be viewed. Image compressors may use this as an indication of the color or grayscale resolution of the compressed images. If you set this parameter to 0, the sequence grabber component determines the appropriate value for the source image. Values of 1, 2, 4, 8, 16, 24, and 32 indicate the number of bits per pixel for color images. Values of 33, 34, 36, and 40 indicate 1-bit, 2-bit, 4-bit, and 8-bit grayscale, respectively, for grayscale images. Your program can determine which depths are supported by a given compressor by examining the compressor information structure returned by the Image Compression Manager's `GetCodecInfo` function (see the chapter "Image Compression Manager" in *Inside Macintosh: QuickTime* for more information on the `GetCodecInfo` function).  
Set this parameter to 0 to leave the depth unchanged.
- `compressor` Specifies the image compressor identifier. Specify a particular compressor by setting this parameter to its compressor identifier. You can obtain this identifier from the Image Compression Manager's `GetCodecNameList` function. Set this parameter to 0 to leave the compressor unchanged.
- `spatialQuality` Specifies the desired compressed image quality. See the chapter "Image Compression Manager" in *Inside Macintosh: QuickTime* for valid values.
- `temporalQuality` Specifies the desired sequence temporal quality. This parameter governs the level of compression you desire with respect to information between successive frames in the sequence. Set this parameter to 0 to prevent the image compressor from applying temporal compression to the sequence. See the chapter "Image Compression Manager" in *Inside Macintosh: QuickTime* for other valid values.
- `keyFrameRate` Specifies the maximum number of frames allowed between key frames. Key frames provide points from which a temporally compressed sequence may be decompressed. Use this parameter to control the



frequency at which the image compressor places key frames into the compressed sequence. For more information about key frames, see the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime*. The compressor determines the optimum placement for key frames based upon the amount of redundancy between adjacent images in the sequence. Consequently, the compressor may insert key frames more frequently than you have requested. However, the compressor will never place key frames less often than is indicated by the setting of the `keyFrameRate` parameter. The compressor ignores this parameter if you have not requested temporal compression (that is, you have set the `temporalQuality` parameter to 0).

DESCRIPTION

Typically, you are interested in setting only one or two of these parameters. You can call the `SGGetVideoCompressor` function to retrieve the values of all of the parameters, and you can then use that information to supply values for the parameters you do not wish to change.

SPECIAL CONSIDERATIONS

You may call this function during a record operation or after you have prepared the sequence grabber component for a record operation only if you set the `depth` and `compressor` parameters to 0. This allows you to work with the quality or key frame rate configuration while you are capturing a sequence.

RESULT CODES

|                                      |       |  |
|--------------------------------------|-------|--|
| <code>paramErr</code>                | -50   | Invalid parameter specified              |
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode          |
| <code>notEnoughMemoryToGrab</code>   | -9403 | Insufficient memory for record operation |
| <code>deviceCantMeetRequest</code>   | -9408 | Device cannot support grabber            |

SGGetVideoCompressor

The `SGGetVideoCompressor` function allows you to determine a channel’s current image-compression parameters.

```
pascal ComponentResult SGGetVideoCompressor (SGChannel c,
                                             short *depth,
                                             compressorComponent *compressor,
                                             CodecQ *spatialQuality,
                                             CodecQ *temporalQuality,
                                             long *keyFrameRate);
```

## Sequence Grabber Components

|                              |   |
|------------------------------|---|
| <code>c</code>               | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.  |
| <code>depth</code>           | <p>Contains a pointer to a field that is to receive the depth at which the image is likely to be viewed. Image compressors may use this as an indication of the color or grayscale resolution of the compressed images. If the value returned by this function is 0, the sequence grabber component determines the appropriate value for the source image. Values of 1, 2, 4, 8, 16, 24, and 32 indicate the number of bits per pixel for color images. Values of 33, 34, 36, and 40 indicate 1-bit, 2-bit, 4-bit, and 8-bit grayscale, respectively, for grayscale images. Your program can determine which depths are supported by a given compressor by examining the compressor information record (defined by the <code>CodecInfo</code> data type) returned by the Image Compression Manager's <code>GetCodecInfo</code> function (see the chapter "Image Compression Manager" in <i>Inside Macintosh: QuickTime</i> for more information on the <code>GetCodecInfo</code> function).</p> <p>If you are not interested in this information, set this parameter to <code>nil</code>.</p> |
| <code>compressor</code>      | <p>Contains a pointer a field that is to receive an image compressor identifier. If you are not interested in this information, set this parameter to <code>nil</code>.</p>   |
| <code>spatialQuality</code>  | <p>Contains a pointer to a field that is to receive the desired compressed image quality. See the chapter "Image Compression Manager" in <i>Inside Macintosh: QuickTime</i> for valid values. If you are not interested in this information, set this parameter to <code>nil</code>.</p>  |
| <code>temporalQuality</code> | <p>Contains a pointer to a field that is to receive the desired sequence temporal quality. This parameter governs the level of compression you desire with respect to information between successive frames in the sequence. If the returned value is set to 0, the image compressor is not performing temporal compression on the source video. See the chapter "Image Compression Manager" in <i>Inside Macintosh: QuickTime</i> for other valid values.</p> <p>If you are not interested in this information, set this parameter to <code>nil</code>.</p>  |
| <code>keyFrameRate</code>    | <p>Contains a pointer to a field that is to receive the maximum number of frames allowed between key frames. Key frames provide points from which a temporally compressed sequence may be decompressed. This value controls the frequency at which the image compressor places key frames into the compressed sequence. The compressor determines the optimum placement for key frames based upon the amount of redundancy between adjacent images in the sequence. Consequently, the compressor may insert key frames more frequently than you have</p>  |

requested. However, the compressor will never place key frames less often than is indicated by the setting of the `keyFrameRate` parameter. The compressor ignores this value if you have not requested temporal compression (that is, you have set the `temporalQuality` parameter of the `SGSetVideoCompressor` function to 0).

If you are not interested in this information, set this parameter to `nil`.

**SEE ALSO**

You can set these parameters by calling the `SGSetVideoCompressor` function, which is described in the previous section.

**SGSetVideoDigitizerComponent**

The `SGSetVideoDigitizerComponent` function allows you to assign a video digitizer component to a video channel.

```
pascal ComponentResult SGSetVideoDigitizerComponent
                        (SGChannel c, ComponentInstance vdig);
```

|                   |   |
|-------------------|---|
| <code>c</code>    | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.  |
| <code>vdig</code> | Contains a component instance that identifies a connection to a video digitizer component. The specified video channel component uses this video digitizer component to obtain its source video data. For more information about video digitizer components, see the chapter “Video Digitizer Components” in this book. |

**DESCRIPTION**

Typically, the video channel component locates its own video digitizer component. Consequently, you may not need to use the `SGSetVideoDigitizerComponent` function.

**SPECIAL CONSIDERATIONS**

You cannot use the `SGSetVideoDigitizerComponent` function during a record operation. Many values are reinitialized as a result of changing digitizers.

**RESULT CODE**

|                                      |                    |                                 |
|--------------------------------------|--------------------|---------------------------------|
| <code>cantDoThatInCurrentMode</code> | <code>-9402</code> | Request invalid in current mode |
|--------------------------------------|--------------------|---------------------------------|

## SGGetVideoDigitizerComponent

---

The `SGGetVideoDigitizerComponent` function allows you to determine the video digitizer component that is providing source video to a video channel component. You can use this function to obtain access to the video digitizer component so that you can set its parameters, if you so desire. See the chapter “Video Digitizer Components” in this book for information about video digitizer components.

```
pascal ComponentInstance SGGetVideoDigitizerComponent
                                (SGChannel c);
```

**c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

### DESCRIPTION

The `SGGetVideoDigitizerComponent` function returns a component instance that identifies the connection between the video channel component and its video digitizer component. If the video channel component does not use a video digitizer component, this returned value is set to `nil`.

### SPECIAL CONSIDERATIONS

If you change any video digitizer component parameters, be sure to notify the sequence grabber component by calling the `SGVideoDigitizerChanged` function, which is described in the next section. In addition, you should not change any video digitizer component parameters during a record operation.

## SGVideoDigitizerChanged

---

The `SGVideoDigitizerChanged` function allows you to notify the sequence grabber component whenever you change the configuration of a video channel’s video digitizer.

```
pascal ComponentResult SGVideoDigitizerChanged (SGChannel c);
```

**c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

### DESCRIPTION

The sequence grabber and its video channels maintain information about the configuration of any video digitizer components that are currently in use.

**IMPORTANT**

It is very important to notify the sequence grabber of any configuration changes you make. ▲

**SPECIAL CONSIDERATIONS**

You should not change the configuration of the video digitizer during a record operation.

**SEE ALSO**

You can obtain access to a video channel's video digitizer component by calling the `SGGetVideoDigitizerComponent` function, which is described in the previous section.

**RESULT CODE**

`cantDoThatInCurrentMode`      -9402      Request invalid in current mode

**SGSetCompressBuffer**

Some video source data may contain unacceptable levels of visual noise or artifacts. One technique for removing this noise is to capture the image and then reduce it in size. During the size reduction process, the noise can be filtered out.

The `SGSetCompressBuffer` function creates a filter buffer for a video channel. Logically, this buffer sits between the source video buffer and the destination rectangle you set with the `SGSetChannelBounds` function, described on page 5-65. The filter buffer should be larger than the area enclosed by the destination rectangle.

```
pascal ComponentResult SGSetCompressBuffer (SGChannel c,
                                             short depth,
                                             const Rect *compressSize);
```

|                           |   |
|---------------------------|---|
| <code>c</code>            | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.  |
| <code>depth</code>        | Specifies the pixel depth of the filter buffer. If you set this parameter to 0, the sequence grabber component uses the depth of the video buffer.  |
| <code>compressSize</code> | Contains a pointer to the dimensions of the filter buffer. This buffer should be larger than the destination buffer. Set this parameter to <code>nil</code> , or set the coordinates of this rectangle to 0 (specifying an empty rectangle), to stop filtering the input source video data. |

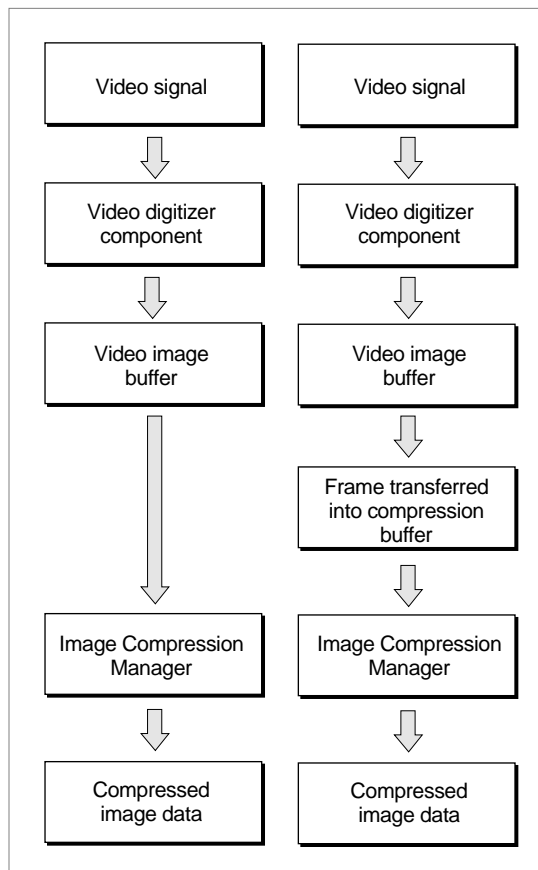
## Sequence Grabber Components

## DESCRIPTION

If you establish a filter buffer for a channel, the sequence grabber component places the captured video image into the filter buffer, then copies the image into the destination buffer. This process may be too slow for some record operations, but can be useful during controlled record operations (where the source video can be read on a frame-by-frame basis). Be sure to call this function before you prepare the sequence grabber component for the record or playback operation.

Figure 5-2 demonstrates the process by which the `SGSetCompressBuffer` function creates a filter buffer for a video channel.

**Figure 5-2** The effect of the `SGSetCompressBuffer` function



## SEE ALSO

If you want to perform some more elaborate image filtering, you may define a transfer-frame function. See “Video Channel Callback Functions” beginning on page 5-99 for more information about transfer-frame functions.

**RESULT CODE**

`cantDoThatInCurrentMode`     **-9402**     Request invalid in current mode

**SGGetCompressBuffer**

---

The `SGGetCompressBuffer` function returns information about the filter buffer you have established for a video channel.

```
pascal ComponentResult SGGetCompressBuffer (SGChannel c,
                                             short *depth,
                                             Rect *compressSize);
```

- |                           |  |
|---------------------------|--|
| <code>c</code>            | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.   |
| <code>depth</code>        | Contains a pointer to a field that is to receive the pixel depth of the filter buffer. If the returned value is set to 0, the sequence grabber is not filtering the input video data.  |
| <code>compressSize</code> | Contains a pointer to a rectangle structure that is to receive the dimensions of the filter buffer. If the sequence grabber is not filtering the input video data, it returns an empty rectangle (all coordinates set to 0). |

**SEE ALSO**

You set a filter buffer by calling the `SGSetCompressBuffer` function, which is described in the previous section.

**SGSetFrameRate**

---

The `SGSetFrameRate` function allows you to specify a video channel's frame rate for recording.

```
pascal ComponentResult SGSetFrameRate (SGChannel c,
                                         Fixed frameRate);
```

- |                        |  |
|------------------------|--|
| <code>c</code>         | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, discussed on page 5-31 and page 5-32, respectively. |
| <code>frameRate</code> | Specifies the desired frame rate. Set this parameter to 0 to select the channel's default frame rate. Typically, this corresponds to the fastest rate that the channel can support.  |

## Sequence Grabber Components

**DESCRIPTION**

The `SGSetFrameRate` function allows you to control a video channel's frame rate. Note that the digitizing hardware may not be able to support the full rate you specify. If you specify too high a rate, the sequence grabber operates at the highest rate that it can support. Note that you may not call this function when you are recording.

**RESULT CODES**

|                                      |       |                                 |
|--------------------------------------|-------|---------------------------------|
| <code>paramErr</code>                | -50   | Invalid parameter value         |
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode |

**SEE ALSO**

You can retrieve a channel's current frame rate by calling the `SGGetFrameRate` function, which is described next.

**SGGetFrameRate**

---

The `SGGetFrameRate` function allows you to retrieve a video channel's frame rate for recording.

```
pascal ComponentResult SGGetFrameRate (SGChannel c,
                                         Fixed *frameRate);
```

|                        |  |
|------------------------|--|
| <code>c</code>         | Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the <code>SGNewChannel</code> or <code>SGNewChannelFromComponent</code> function, discussed on page 5-31 and page 5-32, respectively. |
| <code>frameRate</code> | Contains a pointer to a field to receive the current frame rate. The sequence grabber returns the channel's current frame rate.  |

**DESCRIPTION**

The `SGGetFrameRate` function returns the channel's current rate. By default, the channel records at the fastest rate it can support. In this case, the channel sets the field referred to by the `frameRate` parameter to 0.

**SEE ALSO**

You can set a channel's frame rate by calling the `SGSetFrameRate` function, which is described in the previous section.



## SGSetUseScreenBuffer

---

The `SGSetUseScreenBuffer` function allows you to control whether a video channel uses an offscreen buffer.

```
pascal ComponentResult SGSetUseScreenBuffer (SGChannel c,
                                             Boolean useScreenBuffer);
```

**c** Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the `SGNewChannel` or `SGNewChannelFromComponent` function, discussed on page 5-31 and page 5-32, respectively.

**useScreenBuffer** Indicates whether to use an offscreen buffer. If you set this parameter to `true`, the channel draws directly to the screen. If you set it to `false`, the channel may use an offscreen buffer. If the channel cannot work with offscreen buffers, it ignores this parameter.

### DESCRIPTION

By default, video channels try to draw directly to the screen. The `SGSetUseScreenBuffer` function allows you to direct a video channel to draw to an offscreen buffer. If the channel cannot draw offscreen, it ignores this function. Note that you may not call this function when you are recording.

Directing a channel to draw offscreen may be useful if you are performing transformations on the data before displaying it (such as blending it with another graphical image).

### RESULT CODES

|                                      |       |                                 |
|--------------------------------------|-------|---------------------------------|
| <code>paramErr</code>                | -50   | Invalid parameter value         |
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode |

### SEE ALSO

You can determine whether you have allowed a channel to draw offscreen by calling the `SGGetUseScreenBuffer` function, which is described next.

## SGGetUseScreenBuffer

---

The `SGGetUseScreenBuffer` function allows you to determine whether a video channel is allowed to use an offscreen buffer.

```
pascal ComponentResult SGGetUseScreenBuffer (SGChannel c,
                                             Boolean *useScreenBuffer);
```

## Sequence Grabber Components

**c** Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the `SGNewChannel` or `SGNewChannelFromComponent` function, discussed on page 5-31 and page 5-32, respectively.

**useScreenBuffer**

Contains a pointer to a Boolean value. The sequence grabber sets this field to reflect whether you have allowed the channel to draw offscreen. If this field is set to `true`, the channel draws directly to the screen. If it is set to `false`, the channel may use an offscreen buffer. If the channel cannot work with offscreen buffers, it ignores this value.

**DESCRIPTION**

By default, video channels draw directly to the screen. You can direct a channel to draw to an offscreen buffer by calling the `SGSetUseScreenBuffer` function. Channels that can work offscreen then allocate and draw to an offscreen buffer.

**SEE ALSO**

You can allow a channel to draw offscreen by calling the `SGSetUseScreenBuffer` function, which is described in the previous section.

## Working With Sound Channels

---

Sequence grabber components provide a number of functions that allow you to configure the grabber's sound channels. This section describes these configuration functions, which you can use only with sound channels. You can determine whether a channel has a sound representation by calling the `SGGetChannelInfo` function, described on page 5-61. If you want to configure a video channel, use the functions described in "Working With Video Channels" beginning on page 5-77. If you want to configure general attributes of a channel, use the functions described in "Working With Channel Characteristics" beginning on page 5-58.

Use the `SGSetSoundInputDriver` function to specify a channel's sound input device. You can determine a channel's sound input device by calling the `SGGetSoundInputDriver` function. If you change any attributes of the sound input device, you should notify the sequence grabber component by calling the `SGSoundInputDriverChanged` function. By default, the sequence grabber component uses the sound driver's best settings.

You can control the amount of sound data the sequence grabber works with at one time by calling the `SGSetSoundRecordChunkSize` function. You can determine this value by calling the `SGGetSoundRecordChunkSize` function.

You can control the rate at which the sound channel samples the input data by calling the `SGSetSoundInputRate` function. You can determine the sample rate by calling the `SGGetSoundInputRate` function.

You can control other sound input parameters by using the `SGSetSoundInputParameters` and `SGGetSoundInputParameters` functions.

## SGSetSoundInputDriver

---

Some sound channel components may use sound input devices to obtain their source data. The `SGSetSoundInputDriver` function allows you to assign a sound input device to a sound channel.

```
pascal ComponentResult SGSetSoundInputDriver (SGChannel c,
                                              const Str255 driverName);
```

- `c` Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.
- `driverName` Specifies the name of the sound input device. This is a Pascal string, and it must correspond to a valid sound input device.

### DESCRIPTION

If the sound channel component does not use sound input devices, it returns a nonzero result code. For more information about sound input devices, see *Inside Macintosh: More Macintosh Toolbox*—in particular, refer to the discussion of the Sound Manager’s `SPBGetIndexedDevice` routine.

### SPECIAL CONSIDERATIONS

You cannot call the `SGSetSoundInputDriver` function during a record operation.

### RESULT CODES

|                                      |                    |  |
|--------------------------------------|--------------------|--|
| <code>noDeviceForChannel</code>      | <code>-9400</code> | Channel component cannot find its device |
| <code>cantDoThatInCurrentMode</code> | <code>-9402</code> | Request invalid in current mode          |
| <code>deviceCantMeetRequest</code>   | <code>-9408</code> | Device cannot support grabber            |

## SGGetSoundInputDriver

---

The `SGGetSoundInputDriver` function allows you to determine the sound input device currently in use by a sound channel component.

```
pascal long SGGetSoundInputDriver (SGChannel c);
```

## Sequence Grabber Components

- c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

**DESCRIPTION**

The `SGGetSoundInputDriver` function returns a reference to the sound input device. If the sound channel is not using a sound input device, this returned value is set to `nil`.

You may want to gain access to the sound input device if you want to change the device's configuration.

**SPECIAL CONSIDERATIONS**

If you change any of the device's operating parameters, be sure to inform the sequence grabber component by calling the `SGSoundInputDriverChanged` function, which is described in the next section.

**SEE ALSO**

You can assign a sound input device to a sound channel by calling the `SGSetSoundInputDriver` function, described in the previous section.

## SGSoundInputDriverChanged

---

The `SGSoundInputDriverChanged` function allows you to notify the sequence grabber component whenever you change the configuration of a sound channel's sound input device.

```
pascal ComponentResult SGSoundInputDriverChanged (SGChannel c);
```

- c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

**DESCRIPTION**

The sequence grabber's sound channels maintain information about the configuration of any sound input devices that are currently in use. It is very important to notify the sequence grabber component of any configuration changes you make.

**SPECIAL CONSIDERATIONS**

You should not change the configuration of the sound input device during a record operation.

SEE ALSO

You can obtain access to a sound channel's sound input device by calling the `SGGetSoundInputDriver` function, which is described in the previous section.

SGSetSoundRecordChunkSize

During record operations, the sequence grabber works with groups of sound samples. These groups are referred to as *chunks*. By default, each chunk contains two seconds of sound data. Smaller chunks use less memory. You can control the amount of sound data in each chunk by calling the `SGSetSoundRecordChunkSize` function.

```
pascal ComponentResult SGSetSoundRecordChunkSize (SGChannel c,
                                                    long seconds);
```

- `c` Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.
- `seconds` Specifies the number of seconds of sound data the sequence grabber is to work with at a time. To specify a fraction of a second, set this parameter to a negative fixed-point number. For example, to set the duration to half a second, pass in `-0.5` in this parameter.

DESCRIPTION

You specify the number of seconds of sound data the sequence grabber is to work with at a time.

SPECIAL CONSIDERATIONS

You cannot call the `SGSetSoundRecordChunkSize` function during a record or preview operation, or after you have prepared the sequence grabber for a record or preview operation (by calling the `SGPrepare` function, described on page 5-43). This function may return a fraction (for details, see the discussion of the `seconds` parameter above).

RESULT CODES

|                                      |       |                                 |
|--------------------------------------|-------|---------------------------------|
| <code>paramErr</code>                | -50   | Invalid parameter specified     |
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode |

## SGGetSoundRecordChunkSize

---

The `SGGetSoundRecordChunkSize` function allows you to determine the amount of sound data the sequence grabber component works with at a time.

```
pascal long SGGetSoundRecordChunkSize (SGChannel c);
```

**c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

### DESCRIPTION

`SGGetSoundRecordChunkSize` returns a long integer that specifies the number of seconds of sound data the sequence grabber works with at a time.

### SEE ALSO

You set the amount of sound data the sequence grabber component works with at any given time by calling the `SGSetSoundRecordChunkSize` function, which is described in the previous section.

## SGSetSoundInputRate

---

The `SGSetSoundInputRate` function allows you to set the rate at which the sound channel obtains its sound data.

```
pascal ComponentResult SGSetSoundInputRate (SGChannel c,
                                             Fixed rate);
```

**c** Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

**rate** Specifies the rate at which the sound channel is to acquire data. This parameter specifies the number of samples the sound channel is to generate per second. If the sound channel cannot support the rate you specify, it uses the closest available rate that it supports—you can use the `SGGetSoundInputRate` function, described in the next section, to retrieve the rate being used by the channel. Set this parameter to 0 to cause the sound channel to use its default rate.

You can determine the rates that are valid for a sound channel that uses a sound input device by calling the Sound Manager (see *Inside Macintosh: More Macintosh Toolbox* for more information about the Sound Manager).

**RESULT CODES**

|                                      |       |                                 |
|--------------------------------------|-------|---------------------------------|
| <code>cantDoThatInCurrentMode</code> | -9402 | Request invalid in current mode |
| <code>deviceCantMeetRequest</code>   | -9408 | Device cannot support grabber   |

**SGGetSoundInputRate**

---

The `SGGetSoundInputRate` function allows you to determine the rate at which the sound channel is collecting sound data.

```
pascal Fixed SGGetSoundInputRate (SGChannel c);
```

`c` Specifies the reference that identifies the channel for this operation. You obtain this reference from the `SGNewChannel` function, described on page 5-31.

**DESCRIPTION**

`SGGetSoundInputRate` returns a fixed-point number that indicates the number of samples the sound channel collects per second.

**SEE ALSO**

You set the rate at which the sound channel is collecting data by calling the `SGSetSoundInputRate` function, which is described in the previous section.

**SGSetSoundInputParameters**

---

The `SGSetSoundInputParameters` function allows you to set some parameters that relate to sound recording.

```
pascal ComponentResult SGSetSoundInputParameters (SGChannel c,
                                                    short sampleSize,
                                                    short numChannels,
                                                    OSType compressionType);
```

`c` Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the `SGNewChannel` or `SGNewChannelFromComponent` function, discussed on page 5-31 and page 5-32, respectively.

`sampleSize` Specifies the number of bits in each sound sample. Set this field to 8 for 8-bit sound; set it to 16 for 16-bit sound.

## Sequence Grabber Components

`numChannels`

Indicates the number of sound channels used by the sound sample. Set this field to 1 for monaural sounds; set it to 2 for stereo sounds.

`compressionType`

Describes the format of the sound data. The following values are supported:

|                      |  |
|----------------------|--|
| <code>'raw '</code>  | Sound samples are uncompressed, in offset-binary format (that is, sample data values range from 0 to 255). |
| <code>'MAC3 '</code> | Sound samples have been compressed by the Sound Manager at a ratio of 3:1.                                 |
| <code>'MAC6 '</code> | Sound samples have been compressed by the Sound Manager at a ratio of 6:1.                                 |

## DESCRIPTION

You may use the `SGSetSoundInputParameters` function to control many parameters relating to sound recording. All of the sound parameters support two special values. If you set any of these parameters to 0, the sequence grabber does not change the current value of that parameter. If you set any of them to -1, the sequence grabber returns that parameter to its default value.

If you select a parameter value that the sound device cannot support, the sequence grabber returns an appropriate Sound Manager result code.

## RESULT CODES

Sound Manager errors

**SGGetSoundInputParameters**

---

The `SGGetSoundInputParameters` function allows you to retrieve some parameters that relate to sound recording.

```
pascal ComponentResult SGGetSoundInputParameters (SGChannel c,
                                                    short *sampleSize,
                                                    short *numChannels,
                                                    OSType *compressionType);
```

`c` Identifies the channel for this operation. You provide your connection identifier. You connect to a channel component by calling the `SGNewChannel` or `SGNewChannelFromComponent` function, discussed on page 5-31 and page 5-32, respectively.



## Sequence Grabber Components

`sampleSize`

Contains a pointer to a field to receive the sample size. The sequence grabber sets this field to 8 for 8-bit sound; it sets the field to 16 for 16-bit sound.

`numChannels`

Contains a pointer to a field to receive the number of sound channels used by the sound sample. The sequence grabber sets this field to 1 for monaural sounds; it sets the field to 2 for stereo sounds.

`compressionType`

Contains a pointer to a field that is to receive the format of the sound data. The following values may be returned:

|                     |  |
|---------------------|--|
| <code>'raw'</code>  | Sound samples are uncompressed, in offset-binary format (that is, sample data values range from 0 to 255). |
| <code>'MAC3'</code> | Sound samples have been compressed by the Sound Manager at a ratio of 3:1.                                 |
| <code>'MAC6'</code> | Sound samples have been compressed by the Sound Manager at a ratio of 6:1.                                 |

**DESCRIPTION**

You may use the `SGGetSoundInputParameters` function to retrieve many parameters relating to sound recording. If you set any of the sound parameters to `nil`, the sequence grabber does not return that value.

## Video Channel Callback Functions

Sequence grabber components allow you to define a number of callback functions in your application. The sequence grabber calls your functions at specific points in the process of collecting, compressing, and displaying the source video data. By defining callback functions, you can control the process more precisely or customize the operation of the sequence grabber component.

For example, you could use a callback function to draw a frame number on each video frame as it is collected. You could use either a `compress` callback function or a `grab-complete` callback function to accomplish this. The `compress` callback function is called after each frame is collected, in order to compress the frame. The `grab-complete` callback function is called just before the `compress` callback function, as soon as the frame has been captured.

The `SGSetVideoBottlenecks` function lets you assign callback functions to a video channel. You can use the `SGGetVideoBottlenecks` function to determine the callback functions that have been assigned to a video channel.

The `SGSetVideoBottlenecks` function accepts a video bottlenecks structure that identifies the callback functions to be assigned to the channel. In addition, the `SGGetVideoBottlenecks` function contains a pointer to this structure.

The video bottlenecks structure is defined by the `VideoBottles` data type as follows:

```
struct VideoBottles {
    short          procCount;          /* count of callbacks */
    GrabProc       grabProc;           /* grab function */
    GrabCompleteProc grabCompleteProc; /* grab-complete function */
    DisplayProc    displayProc;        /* display function */
    CompressProc   compressProc;       /* compress function */
    CompressCompleteProc compressCompleteProc; /* compress-complete
                                           function */
    AddFrameProc   addFrameProc;       /* add-frame function */
    TransferFrameProc transferFrameProc; /* transfer-frame function */
    GrabCompressCompleteProc grabCompressCompleteProc; /* grab-compress-complete
                                                         function */
    DisplayCompressProc displayCompressProc; /* display-compress
                                              function */
};
typedef struct VideoBottles VideoBottles;
```

#### Field descriptions

|                                   |   |
|-----------------------------------|---|
| <code>procCount</code>            | Specifies the number of callback functions that may be identified in the structure. Set this field to 9.  |
| <code>grabProc</code>             | Identifies the grab function. If you are setting a grab function, set this field so that it points to the function's entry point. If you are not setting a grab function, set this field to <code>nil</code> .  |
| <code>grabCompleteProc</code>     | Identifies the grab-complete function. If you are setting a grab-complete function, set this field so that it points to the function's entry point. If you are not setting a grab-complete function, set this field to <code>nil</code> .             |
| <code>displayProc</code>          | Identifies the display function. If you are setting a display function, set this field so that it points to the function's entry point. If you are not setting a display function, set this field to <code>nil</code> .                               |
| <code>compressProc</code>         | Identifies the compress function. If you are setting a compress function, set this field so that it points to the function's entry point. If you are not setting a compress function, set this field to <code>nil</code> .                            |
| <code>compressCompleteProc</code> | Identifies the compress-complete function. If you are setting a compress-complete function, set this field so that it points to the function's entry point. If you are not setting a compress-complete function, set this field to <code>nil</code> . |

## Sequence Grabber Components

|                                       |  |
|---------------------------------------|--|
| <code>addFrameProc</code>             | Identifies the add-frame function. If you are setting an add-frame function, set this field so that it points to the function's entry point. If you are not setting an add-frame function, set this field to <code>nil</code> .                                      |
| <code>transferFrameProc</code>        | Identifies the transfer-frame function. If you are setting a transfer-frame function, set this field so that it points to the function's entry point. If you are not setting a transfer-frame function, set this field to <code>nil</code> .                         |
| <code>grabCompressCompleteProc</code> | Identifies the grab-compress-complete function. If you are setting a grab-compress-complete function, set this field so that it points to the function's entry point. If you are not setting a grab-compress-complete function, set this field to <code>nil</code> . |
| <code>displayCompressProc</code>      | Identifies the display-compress function. If you are setting a display-compress function, set this field so that it points to the function's entry point. If you are not setting a display-compress function, set this field to <code>nil</code> .                   |

## SGSetVideoBottlenecks

---

The `SGSetVideoBottlenecks` function assigns callback functions to a video channel.

```
pascal ComponentResult SGSetVideoBottlenecks (SGChannel c,
                                              VideoBottles *vb);
```

|                 |  |
|-----------------|--|
| <code>c</code>  | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.   |
| <code>vb</code> | Contains a pointer to a video bottlenecks structure (defined by the <code>VideoBottles</code> data type). That structure identifies the callback functions to be assigned to this video channel. The video bottlenecks structure is described on page 5-100. |

### DESCRIPTION

The `SGSetVideoBottlenecks` function accepts a video bottlenecks structure that identifies the callback functions to be assigned to the channel.

### SPECIAL CONSIDERATIONS

Your application should not call this function during a record or playback operation.

## SGGetVideoBottlenecks

---

The `SGGetVideoBottlenecks` function allows you to determine the callback functions that have been assigned to a video channel.

```
pascal ComponentResult SGGetVideoBottlenecks (SGChannel c,
                                              VideoBottles *vb);
```

- |                 |  |
|-----------------|--|
| <code>c</code>  | Specifies the reference that identifies the channel for this operation. You obtain this reference from the <code>SGNewChannel</code> function, described on page 5-31.   |
| <code>vb</code> | Contains a pointer to a video bottlenecks structure, described on page 5-100. The <code>SGGetVideoBottlenecks</code> function sets the fields of that structure to indicate the callback functions that have been assigned to this video channel. You must set the <code>procCount</code> field in the video bottlenecks structure to 9. |

### SEE ALSO

You assign callback functions to a video channel by calling the `SGSetVideoBottlenecks` function, which is described in the previous section.

## Utility Functions for Video Channel Callback Functions

---

Sequence grabber components provide a number of functions that your callback functions can use. This section describes those functions.

Use the `SGGetBufferInfo` function to obtain information about a buffer that contains data to be manipulated by your callback function.

The remaining functions described here provide default behavior for your callback functions.

## SGGetBufferInfo

---

You can use the `SGGetBufferInfo` function to obtain information about a buffer that has been passed to your callback function.

```
pascal ComponentResult SGGetBufferInfo (SGChannel c,
                                         short bufferNum,
                                         PixMapHandle *bufferPM,
                                         Rect *bufferRect,
                                         GWorldPtr *compressBuffer,
                                         Rect *compressBufferRect);
```

## Sequence Grabber Components

|                                 |  |
|---------------------------------|--|
| <code>c</code>                  | Specifies the reference that identifies the channel for this operation.  |
| <code>bufferNum</code>          | Identifies the buffer. The sequence grabber component provides this value to your callback function.   |
| <code>bufferPM</code>           | Contains a pointer to a location that is to receive a handle to the pixel map that contains the image. Note that this pixel map may be offscreen. Do not dispose of this pixel map. If you do not want this information, set this parameter to <code>nil</code> .  |
| <code>bufferRect</code>         | Contains a pointer to a rectangle structure that is to receive the dimensions of the image's boundary rectangle. If you do not want this information, set this parameter to <code>nil</code> .   |
| <code>compressBuffer</code>     | <p>Contains a pointer to a location that is to receive a pointer to the filter buffer for the image. The sequence grabber component returns this information only if your application has assigned a filter buffer to this video channel. You assign a filter buffer by calling the <code>SGSetCompressBuffer</code> function, which is described on page 5-87. Do not dispose of this buffer.</p> <p>If you have not assigned a filter buffer, the sequence grabber sets the returned value to <code>nil</code>. If you do not want this information, set this parameter to <code>nil</code>.</p> |
| <code>compressBufferRect</code> | <p>Contains a pointer to a rectangle structure that is to receive the dimensions of the filter buffer for the image. The sequence grabber component returns this information only if your application has assigned a filter buffer to this video channel. You assign a filter buffer by calling the <code>SGSetCompressBuffer</code> function, which is described on page 5-87. If you have not assigned a filter buffer, the sequence grabber component returns an empty rectangle. If you do not want this information, set this parameter to <code>nil</code>.</p>                              |

## RESULT CODE

|                       |     |                             |
|-----------------------|-----|-----------------------------|
| <code>paramErr</code> | -50 | Invalid parameter specified |
|-----------------------|-----|-----------------------------|

**SGGrabFrame**

The `SGGrabFrame` function provides the default behavior for your grab function.

```
pascal ComponentResult SGGrabFrame (SGChannel c, short bufferNum);
```

|                        |   |
|------------------------|---|
| <code>c</code>         | Specifies the reference that identifies the channel for this operation. The sequence grabber component provides this value to your grab function. |
| <code>bufferNum</code> | Identifies the buffer. The sequence grabber component provides this value to your grab function.  |

**SPECIAL CONSIDERATIONS**

You should call the `SGGrabFrame` function only from your grab function. If you call it at any other time, results are unpredictable.

**SEE ALSO**

See “Application-Defined Functions,” which begins on page 5-111, for information about grab-complete functions.

**RESULT CODE**

`cantDoThatInCurrentMode`      -9402      Request invalid in current mode

**SGGrabFrameComplete**

---

The `SGGrabFrameComplete` function provides the default behavior for your grab-complete function.

```
pascal ComponentResult SGGrabFrameComplete (SGChannel c,
                                             short bufferNum,
                                             Boolean *done);
```

|                        |  |
|------------------------|--|
| <code>c</code>         | Specifies the reference that identifies the channel for this operation. The sequence grabber provides this value to your grab-complete function.   |
| <code>bufferNum</code> | Identifies the buffer. The sequence grabber provides this value to your grab-complete function.  |
| <code>done</code>      | Contains a pointer to a Boolean value. The <code>SGGrabFrameComplete</code> function sets this Boolean value to indicate whether the frame has been completely captured. The function sets the Boolean value to <code>true</code> if the capture is complete, and sets it to <code>false</code> if the capture is incomplete. The sequence grabber provides this pointer to your grab-complete function. |

**SPECIAL CONSIDERATIONS**

You should call the `SGGrabFrameComplete` function only from your grab-complete function. If you call it at any other time, results are unpredictable.

**RESULT CODE**

`cantDoThatInCurrentMode`      -9402      Request invalid in current mode

**SEE ALSO**

See “Application-Defined Functions,” which begins on page 5-111, for details about grab-complete functions.

## SGDisplayFrame

---

The `SGDisplayFrame` function provides the default behavior for your display function.

```
pascal ComponentResult SGDisplayFrame (SGChannel c,
                                       short bufferNum,
                                       MatrixRecord *mp,
                                       RgnHandle clipRgn);
```

|                        |   |
|------------------------|---|
| <code>c</code>         | Specifies the reference that identifies the channel for this operation. The sequence grabber component provides this value to your display function.  |
| <code>bufferNum</code> | Identifies the buffer. The sequence grabber component provides this value to your display function.   |
| <code>mp</code>        | Contains a pointer to a transformation matrix for the display operation. If there is no matrix for the operation, set this parameter to <code>nil</code> .  |
| <code>clipRgn</code>   | Contains a handle to the clipping region for the destination image. This region is defined in the destination coordinate system. If there is no clipping region, set this parameter to <code>nil</code> . |

### SPECIAL CONSIDERATIONS

You should call the `SGDisplayFrame` function only from your display function. If you call it at any other time, results are unpredictable.

### RESULT CODE

`cantDoThatInCurrentMode`     -9402     Request invalid in current mode

### SEE ALSO

See “Application-Defined Functions,” which begins on page 5-111, for details about display functions.

## SGCompressFrame

---

The `SGCompressFrame` function provides the default behavior for your compress function.

```
pascal ComponentResult SGCompressFrame (SGChannel c,
                                       short bufferNum);
```

|                |   |
|----------------|---|
| <code>c</code> | Specifies the reference that identifies the channel for this operation. The sequence grabber provides this value to your compress function. |
|----------------|---|

## Sequence Grabber Components

`bufferNum` Identifies the buffer. The sequence grabber provides this value to your compress function.

**SPECIAL CONSIDERATIONS**

You should call the `SGCompressFrame` function only from your compress function. If you call it at any other time, results are unpredictable.

**RESULT CODES**

`cantDoThatInCurrentMode`    -9402    Request invalid in current mode  
Image Compression Manager errors

**SEE ALSO**

See “Application-Defined Functions,” which begins on page 5-111, for information about compress functions.

**SGCompressFrameComplete**

---

The `SGCompressFrameComplete` function provides the default behavior for your compress-complete function.

```
pascal ComponentResult SGCompressFrameComplete (SGChannel c,
                                                short bufferNum,
                                                Boolean *done,
                                                SGCompressInfo *ci);
```

`c` Specifies the reference that identifies the channel for this operation. The sequence grabber component provides this value to your compress-complete function.

`bufferNum` Identifies the buffer. The sequence grabber component provides this value to your compress-complete function.

`done` Contains a pointer to a Boolean value. The `SGCompressFrameComplete` function sets this Boolean value to indicate whether the frame has been completely compressed. The function sets the Boolean value to `true` if the compression is complete; it sets the Boolean value to `false` if the operation is incomplete. The sequence grabber component provides this pointer to your compress-complete function.



## Sequence Grabber Components

**ci** Contains a pointer to a compression information structure (defined by the `SGCompressInfo` data type). If the compression is complete, the function completely formats this structure with information that is appropriate to the frame just compressed. See “The Compression Information Structure” beginning on page 5-22 for a description of this structure. The sequence grabber component provides this pointer to your compress-complete function.

**SPECIAL CONSIDERATIONS**

You should call the `SGCompressFrameComplete` function only from your compress-complete function. If you call it at any other time, results are unpredictable.

**RESULT CODES**

`cantDoThatInCurrentMode`    -9402    Request invalid in current mode  
Image Compression Manager errors

**SEE ALSO**

See “Application-Defined Functions,” which begins on page 5-111, for information about compress-complete functions.

**SGAddFrame**

The `SGAddFrame` function provides the default behavior for your add-frame function.

```
pascal ComponentResult SGAddFrame (SGChannel c, short bufferNum,
                                   TimeValue atTime,
                                   TimeScale scale,
                                   const SGCompressInfo *ci);
```

**c** Specifies the reference that identifies the channel for this operation. The sequence grabber component provides this value to your add-frame function.

**bufferNum** Identifies the buffer. The sequence grabber component provides this value to your add-frame function.

**atTime** Specifies the time at which the frame was captured, in the time scale specified by the `scale` parameter. The sequence grabber component provides this value to your add-frame function. Your add-frame function can change this value before calling the `SGAddFrame` function. You can determine the duration of a frame by subtracting its capture time from the capture time of the next frame in the sequence.

**scale** Specifies the time scale of the movie. The sequence grabber component provides this value to your add-frame function.

## Sequence Grabber Components

**ci** Contains a pointer to a compression information structure (defined by the `SGCompressInfo` data type). This structure contains information describing the compression characteristics of the image to be added to the movie. See “The Compression Information Structure” beginning on page 5-22 for a description of this structure. The sequence grabber component provides this structure to your add-frame function.

**SPECIAL CONSIDERATIONS**

You should call the `SGAddFrame` function only from your add-frame function. If you call it at any other time, results are unpredictable.

**RESULT CODES**

`cantDoThatInCurrentMode`    -9402    Request invalid in current mode  
Memory Manager errors

**SEE ALSO**

See “Application-Defined Functions,” which begins on page 5-111, for information about add-frame functions.

**SGTransferFrameForCompress**

---

The `SGTransferFrameForCompress` function provides the default behavior for your transfer-frame function.

```
pascal ComponentResult SGTransferFrameForCompress (SGChannel c,
                                                    short bufferNum,
                                                    MatrixRecord *mp,
                                                    RgnHandle clipRgn);
```

**c** Specifies the reference that identifies the channel for this operation. The sequence grabber component provides this value to your transfer-frame function.

**bufferNum** Identifies the buffer. The sequence grabber component provides this value to your transfer-frame function.

**mp** Contains a pointer to a transformation matrix for the transfer operation. If there is no matrix for the operation, set this parameter to `nil`.

**clipRgn** Contains a handle to the clipping region for the destination image. This region is defined in the destination coordinate system. If there is no clipping region, set this parameter to `nil`.

**SPECIAL CONSIDERATIONS**

You should call the `SGTransferFrameForCompress` function only from your transfer-frame function. If you call it at any other time, results are unpredictable.

**RESULT CODE**

`cantDoThatInCurrentMode`     **-9402**     Request invalid in current mode

**SEE ALSO**

See “Application-Defined Functions,” which begins on page 5-111, for information about transfer-frame functions.

## SGGrabCompressComplete

---

The `SGGrabCompressComplete` function provides the default behavior for your grab-compress-complete function.

```
pascal ComponentResult SGGrabCompressComplete (SGChannel c,
                                                Boolean *done,
                                                SGCompressInfo *ci,
                                                TimeRecord *tr);
```

|                   |  |
|-------------------|--|
| <code>c</code>    | Identifies the channel for this operation. The sequence grabber provides this value to your grab-compress-complete function.   |
| <code>done</code> | Contains a pointer to a Boolean value. The <code>SGGrabCompressComplete</code> function sets this value to <code>true</code> when it is done; it sets it to <code>false</code> if the operation is incomplete. The sequence grabber provides this pointer to your grab-compress-complete function.   |
| <code>ci</code>   | Contains a pointer to a compression information structure. When the operation is complete, the <code>SGGrabCompressComplete</code> function fills in this structure with information about the compression operation. The format and content of this structure are discussed earlier in this chapter, beginning on page 5-22.<br><br>The sequence grabber provides this pointer to your grab-compress-complete function. |
| <code>tr</code>   | Contains a pointer to a time record. When the operation is complete, the <code>SGGrabCompressComplete</code> function uses this structure to indicate when the frame was grabbed. The format and content of this structure are discussed in the chapter “Movie Toolbox” in <i>Inside Macintosh: QuickTime</i> .<br><br>The sequence grabber provides this pointer to your grab-compress-complete function.               |

**SPECIAL CONSIDERATIONS**

You should call the `SGGrabCompressComplete` function only from your `grab-compress-complete` function. If you call it at other times, results are unpredictable.

**RESULT CODE**

`cantDoThatInCurrentMode`     **-9402**     Request invalid in current mode

**SEE ALSO**

See “Application-Defined Functions” beginning on page 5-111 for information about `grab-compress-complete` functions.

## SGDisplayCompress

---

The `SGDisplayCompress` function provides the default behavior for your `display-compress` function.

```
pascal ComponentResult SGDisplayCompress (SGChannel c,
                                           Ptr dataPtr,
                                           ImageDescriptionHandle desc,
                                           MatrixRecord *mp,
                                           RgnHandle clipRgn);
```

|                      |  |
|----------------------|--|
| <code>c</code>       | Identifies the channel for this operation. The sequence grabber provides this value to your <code>display-compress</code> function.  |
| <code>dataPtr</code> | Contains a pointer to the compressed image data. The sequence grabber provides this pointer to your <code>display-compress</code> function.  |
| <code>desc</code>    | Specifies a handle to the image description structure to use for the decompression operation. The sequence grabber provides this handle to your <code>display-compress</code> function.                                |
| <code>mp</code>      | Contains a pointer to a matrix structure. This matrix structure contains the transformation matrix to use when displaying the image. If there is no matrix for the operation, set this parameter to <code>nil</code> . |
| <code>clipRgn</code> | Contains a handle to the clipping region for the destination image. This region is defined in the destination coordinate system. If there is no clipping region, set this parameter to <code>nil</code> .              |

**SPECIAL CONSIDERATIONS**

You should call the `SGDisplayCompress` function only from your display-compress function. If you call it at other times, results are unpredictable.

**RESULT CODE**

`cantDoThatInCurrentMode`    **-9402**    Request invalid in current mode

**SEE ALSO**

See the next section, “Application-Defined Functions,” for information about display-compress functions.

## Application-Defined Functions

---

This section describes the functions that your application may supply to sequence grabber components.

Your grab function is used by the sequence grabber component to begin the capture of a frame of video data. Your grab-complete function allows the sequence grabber component to determine whether the current frame-capture operation is complete.

Your display function enables the sequence grabber component to move a captured video image in an offscreen buffer into the destination buffer for the video channel.

The sequence grabber component uses your compress function to commence the compression of a captured video image. Your compress-complete function helps the sequence grabber component to find out if the current frame-compression operation is finished.

Your add-frame function lets the sequence grabber component add a frame to a movie.

The sequence grabber component uses your transfer-frame function to move a video frame from the capture buffer into the channel’s filter buffer.

You may provide two functions for use with compressed-source devices. Your grab-compress-complete function determines when the current capture and compress operation is complete. Your display-compress function decompresses and displays a frame.

The sequence grabber calls your data function whenever any of the grabber’s channels write data to the movie file.

If you call the `SGSettingsDialog` function, described on page 5-48, you must supply a modal-dialog filter function. The interface that your function must provide is discussed on page 5-122.

## MyGrabFunction

---

The sequence grabber component calls your grab function in order to start capturing a frame of video data.

Your grab function must present the following interface:

```
pascal ComponentResult MyGrabFunction (SGChannel c,
                                       short bufferNum,
                                       long refCon);
```

|           |  |
|-----------|--|
| c         | Specifies the reference that identifies the channel for this operation.  |
| bufferNum | Identifies the buffer for this operation. You can obtain information about this buffer by calling the <code>SGGetBufferInfo</code> function, which is described on page 5-102. |
| refCon    | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.                          |

### RESULT CODE

|                         |       |                                 |
|-------------------------|-------|---------------------------------|
| cantDoThatInCurrentMode | -9402 | Request invalid in current mode |
|-------------------------|-------|---------------------------------|

### SEE ALSO

Your grab function can use the sequence grabber component's `SGGrabFrame` function to support the default behavior. `SGGrabFrame` is described on page 5-103.

## MyGrabCompleteFunction

---

The sequence grabber component calls your grab-complete function in order to determine whether the current frame-capture operation is complete. Once a frame has been completely captured, you can modify its contents to suit your needs. For example, you can overlay text onto the video image.

Your function must present the following interface:

```
pascal ComponentResult MyGrabCompleteFunction (SGChannel c,
                                              short bufferNum,
                                              Boolean *done,
                                              long refCon);
```

|           |  |
|-----------|--|
| c         | Specifies the reference that identifies the channel for this operation.  |
| bufferNum | Identifies the buffer for this operation. You can obtain information about this buffer by calling the <code>SGGetBufferInfo</code> function, which is described on page 5-102. |

## Sequence Grabber Components

|        |   |
|--------|---|
| done   | Contains a pointer to a Boolean value. Your function sets this Boolean value to indicate whether the frame has been completely captured. Set the Boolean value to <code>true</code> if the capture is complete; set it to <code>false</code> if it is incomplete. |
| refCon | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.   |

## RESULT CODE

`cantDoThatInCurrentMode`     -9402     Request invalid in current mode

## SEE ALSO

Your grab-complete function can use the sequence grabber component's `SGGrabFrameComplete` function to support the default behavior. `SGGrabFrameComplete` is described on page 5-104.

See Listing 5-6 on page 5-20 for a sample grab-complete function. This function draws the letters “QT” over each video frame in the sequence.

## MyDisplayFunction

---

The sequence grabber component calls your display function in order to transfer a captured video image in an offscreen buffer into the destination buffer for the video channel.

Your display function must support the following interface:

```
pascal ComponentResult MyDisplayFunction (SGChannel c,
                                           short bufferNum,
                                           MatrixRecord *mp,
                                           RgnHandle clipRgn,
                                           long refCon);
```

|           |  |
|-----------|--|
| c         | Specifies the reference that identifies the channel for this operation.  |
| bufferNum | Identifies the buffer for this operation. You can obtain information about this buffer by calling the <code>SGGetBufferInfo</code> function, which is described on page 5-102.   |
| mp        | Contains a pointer to a transformation matrix for the display operation. If there is no matrix for the operation, this parameter is set to <code>nil</code> .  |
| clipRgn   | Contains a handle to the clipping region for the destination image. This region is defined in the destination coordinate system. Apply the clipping region after applying the transformation matrix. If there is no clipping region, this parameter is set to <code>nil</code> . |
| refCon    | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.  |

## Sequence Grabber Components

## RESULT CODE

`cantDoThatInCurrentMode`    -9402    Request invalid in current mode

## SEE ALSO

Your application sets the destination buffer by calling the `SGSetChannelBounds` function, which is described on page 5-65.

Your display function can use the sequence grabber component's `SGDisplayFrame` function to support the default behavior. `SGDisplayFrame` is described on page 5-105.

## MyCompressFunction

---

The sequence grabber component calls your compress function in order to start compressing the captured video image.

Your compress function must support the following interface:

```
pascal ComponentResult MyCompressFunction (SGChannel c,
                                           short bufferNum,
                                           long refCon);
```

|                        |  |
|------------------------|--|
| <code>c</code>         | Specifies the reference that identifies the channel for this operation.  |
| <code>bufferNum</code> | Identifies the buffer for this operation. You can obtain information about this buffer by calling the <code>SGGetBufferInfo</code> function, which is described on page 5-102. |
| <code>refCon</code>    | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.                          |

## RESULT CODES

`cantDoThatInCurrentMode`    -9402    Request invalid in current mode

Image Compression Manager errors

## SEE ALSO

Your compress function can use the sequence grabber component's `SGCompressFrame` function to support the default behavior. `SGCompressFrame` is described on page 5-105. This function uses the Image Compression Manager to compress the video image. For more on the Image Compression Manager, see *Inside Macintosh: QuickTime*.



## MyCompressCompleteFunction

The sequence grabber component calls your compress-complete function in order to determine whether the current frame-compression operation is complete.

Your compress-complete function must support the following interface:

```
pascal ComponentResult MyCompressCompleteFunction (SGChannel c,
                                                    short bufferNum,
                                                    Boolean *done,
                                                    SGCompressInfo *ci,
                                                    long refCon);
```

|           |   |
|-----------|---|
| c         | Specifies the reference that identifies the channel for this operation.   |
| bufferNum | Identifies the buffer for this operation. You can obtain information about this buffer by calling the <code>SGGetBufferInfo</code> function, which is described on page 5-102.  |
| done      | Contains a pointer to a Boolean value. Your function sets this Boolean value to indicate whether the frame has been completely compressed. Set the Boolean value to <code>true</code> if the compression is complete; set it to <code>false</code> if it is incomplete.   |
| ci        | Contains a pointer to a compression information structure (defined by the <code>SGCompressInfo</code> data type). If the compression is complete, your function must completely format this structure with information that is appropriate to the frame just compressed. See “The Compression Information Structure” beginning on page 5-22, for a description of this structure. |
| refCon    | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.   |

### DESCRIPTION

Once a frame has been completely compressed, you can add it to the movie.

### SEE ALSO

Your compress-complete function can use the sequence grabber component’s `SGCompressFrameComplete` function to support the default behavior. `SGCompressFrameComplete` is described on page 5-106.

### RESULT CODES

`cantDoThatInCurrentMode`     -9402     Request invalid in current mode  
Image Compression Manager errors

## MyAddFrameFunction

---

The sequence grabber component calls your add-frame function in order to add a frame to a movie. Your add-frame function must support the following interface:

|   |  |
|---|--|
| <pre>pascal ComponentResult MyAddFrameFunction (SGChannel c,  short bufferNum,  TimeValue atTime,  TimeScale scale,  SGCompressInfo *ci,  long refCon);</pre> |  |
| c   | Specifies the reference that identifies the channel for this operation.  |
| bufferNum   | Identifies the buffer for this operation. You can obtain information about this buffer by calling the <code>SGGetBufferInfo</code> function, which is described on page 5-102.   |
| atTime  | Specifies the time at which the frame was captured, in the time scale specified by the <code>scale</code> parameter. Your add-frame function can change this value before adding the frame to the movie or before calling the <code>SGAddFrame</code> function, which is described on page 5-107. You can determine the duration of a frame by subtracting its capture time from the capture time of the next frame in the sequence. |
| scale   | Specifies the time scale of the movie. You must not change this value.   |
| ci  | Contains a pointer to a compression information structure (defined by the <code>SGCompressInfo</code> data type). This structure contains information describing the compression characteristics of the image to be added to the movie. See “The Compression Information Structure” beginning on page 5-22 for a description of this structure.  |
| refCon  | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.  |

### DESCRIPTION

You can use your add-frame function to modify the contents of the frame before it is added to the movie. This can be useful if you want to place frame numbers onto frames you are recording.

### RESULT CODES

|                         |       |                                 |
|-------------------------|-------|---------------------------------|
| cantDoThatInCurrentMode | -9402 | Request invalid in current mode |
| Memory Manager errors   |       |                                 |

SEE ALSO

Your add-frame function can use the sequence grabber component's `SGAddFrame` function to support the default behavior. `SGAddFrame` is described on page 5-107.

MyTransferFrameFunction

The sequence grabber component calls your transfer-frame function in order to move a video frame from the capture buffer into the channel's filter buffer.

Your transfer-frame function must support the following interface:

```
pascal ComponentResult MyTransferFrameFunction (SGChannel c,
                                                short bufferNum,
                                                MatrixRecord *mp,
                                                RgnHandle clipRgn,
                                                long refCon);
```

|           |  |
|-----------|--|
| c         | Specifies the reference that identifies the channel for this operation.  |
| bufferNum | Identifies the buffer for this operation. You can obtain information about this buffer by calling the <code>SGGetBufferInfo</code> function, which is described on page 5-102.   |
| mp        | Contains a pointer to a transformation matrix for the transfer operation. If there is no matrix for the operation, this parameter is set to <code>nil</code> .   |
| clipRgn   | Contains a handle to the clipping region for the destination image. This region is defined in the destination coordinate system. Apply the clipping region after applying the transformation matrix. If there is no clipping region, this parameter is set to <code>nil</code> . |
| refCon    | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.  |

DESCRIPTION

The sequence grabber component calls this function only when you are filtering the video data. By filtering the video data through a filter buffer, you can eliminate some visual artifacts that result from noisy input video sources. Your application sets a filter buffer by calling the `SGSetCompressBuffer` function, which is described on page 5-87.

If you are using a grab-complete function to determine when frames have been grabbed, you should also implement a grab-compress-complete function (described in the next section). Otherwise, the channel will decompress the specified image before calling your grab-complete function, which will result in significantly lower performance. For details on grab-complete functions, see page 5-112.

**RESULT CODE**

`cantDoThatInCurrentMode`      -9402      Request invalid in current mode

**SEE ALSO**

Your transfer-frame function can use the sequence grabber component's `SGTransferFrameForCompress` function to support the default behavior—`SGTransferFrameForCompress` is described on page 5-108.

## MyGrabCompressCompleteFunction

---

The sequence grabber calls your grab-compress-complete function when it is working with a video digitizer that supports compressed source data. Your grab-compress-complete function is responsible for determining whether the current compressed frame has been completely captured and compressed, essentially combining your grab-complete, compress, and compress-complete functions into one function.

Your function must support the following interface:

```
pascal ComponentResult MyGrabCompressCompleteFunction
                                (SGChannel c,
                                Boolean *done,
                                SGCompressInfo *ci,
                                TimeRecord *tr,
                                long refCon);
```

|                     |   |
|---------------------|---|
| <code>c</code>      | Identifies the channel for this operation.  |
| <code>done</code>   | Contains a pointer to a Boolean value. Set this Boolean value to indicate whether you are finished. Set it to <code>true</code> when you are done; set it to <code>false</code> if the operation is incomplete.   |
| <code>ci</code>     | Contains a pointer to a compression information structure. When the operation is complete, fill in this structure with information about the compression operation. The format and content of this structure are discussed earlier in this chapter, beginning on page 5-22.       |
| <code>tr</code>     | Contains a pointer to a time record. When the operation is complete, fill in this structure with information indicating when the frame was grabbed. The format and content of this structure are discussed in the chapter “Movie Toolbox” in <i>Inside Macintosh: QuickTime</i> . |
| <code>refCon</code> | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.   |

**RESULT CODE**

`cantDoThatInCurrentMode`     -9402     Request invalid in current mode

**SEE ALSO**

Your `grab-compress-complete` function may use the sequence grabber's `SGGrabCompressComplete` function to support the default behavior. `SGGrabCompressComplete` is discussed beginning on page 5-109.

## MyDisplayCompressFunction

---

The sequence grabber calls your display-compress function when it is working with a video digitizer component that supports compressed source data. Your display-compress function is responsible for decompressing and displaying a compressed image.

```
pascal ComponentResult MyDisplayCompressFunction (SGChannel c,
                                                Ptr dataPtr,
                                                ImageDescriptionHandle desc,
                                                MatrixRecord *mp,
                                                RgnHandle clipRgn,
                                                long refCon);
```

|                      |   |
|----------------------|---|
| <code>c</code>       | Identifies the channel for this operation. The sequence grabber provides this value to your display-compress function.  |
| <code>dataPtr</code> | Contains a pointer to the compressed image data.  |
| <code>desc</code>    | Specifies a handle to the image description structure to use for the decompression operation. See the chapter “Image Compression Manager” in <i>Inside Macintosh: QuickTime</i> for more information about this data structure.   |
| <code>mp</code>      | Contains a pointer to a matrix structure. This matrix structure contains the transformation matrix to use when displaying the image. If there is no matrix for the operation, this parameter is set to <code>nil</code> .   |
| <code>clipRgn</code> | Contains a handle to the clipping region for the destination image. This region is defined in the destination coordinate system. Apply the clipping region after the transformation matrix. If there is no clipping region, this parameter is set to <code>nil</code> . |
| <code>refCon</code>  | Contains a reference constant value. You can set this value by calling the <code>SGSetChannelRefCon</code> function, which is described on page 5-67.   |

## Sequence Grabber Components

## RESULT CODE

`cantDoThatInCurrentMode`      -9402      Request invalid in current mode

## SEE ALSO

Your display-compress function may use the sequence grabber's `SGDisplayCompress` function to support the default behavior. `SGDisplayCompress` is discussed beginning on page 5-110.

## MyDataFunction

---

The sequence grabber calls your data function whenever any of the grabber's channels write digitized data to the destination movie file. You assign a data function to the sequence grabber by calling the `SGSetDataProc` function, which is discussed on page 5-35.

Your data function must support the following interface:

```
pascal OSErr MyDataFunction (SGChannel c, Ptr p, long len,
                             long *offset, long chRefCon,
                             TimeValue time, short writeType,
                             long refCon);
```

|                       |  |
|-----------------------|--|
| <code>c</code>        | Identifies the channel component that is writing the digitized data.   |
| <code>p</code>        | Contains a pointer to the digitized data.  |
| <code>len</code>      | Indicates the number of bytes of digitized data.   |
| <code>offset</code>   | Contains a pointer to a field that may specify where you are to write the digitized data, and that is to receive a value indicating where you wrote the data. You must update the field referred to by this parameter, supplying the value indicated by the <code>writeType</code> parameter.  |
| <code>chRefCon</code> | Contains control information. The low-order 16 bits contain sample flags for use by the Movie Toolbox's <code>AddMediaSample</code> function (see the chapter "Movie Toolbox" in <i>Inside Macintosh: QuickTime</i> for information about these flags). The sequence grabber sets these flags as appropriate. The high-order 16 bits are reserved for Apple and are always set to 0. |
| <code>time</code>     | Identifies the starting time of the data, in the channel's time scale. You may use the <code>SGGetChannelTimeScale</code> function to retrieve the channel's time scale (discussed on page 5-68).  |

## Sequence Grabber Components

|                        |  |
|------------------------|--|
| <code>writeType</code> | Indicates the type of write operation being performed. The following values are defined: <ul style="list-style-type: none"> <li><code>seqGrabWriteAppend</code><br/>Append the new data to the end of the file. Set the field referred to by the <code>offset</code> parameter to reflect the location at which you added the data.</li> <li><code>seqGrabWriteReserve</code><br/>Do not write any data to the output file. Instead, reserve space in the output file for the amount of data indicated by the <code>len</code> parameter. Set the field referred to by the <code>offset</code> parameter to the location of the reserved space.</li> <li><code>seqGrabWriteFill</code><br/>Write the data into the location specified by the field referred to by the <code>offset</code> parameter. Set that field to the location of the byte following the last byte you wrote.<br/><br/>This option is used to fill the space reserved previously when the <code>writeType</code> parameter was set to <code>seqGrabWriteReserve</code>. Note that the sequence grabber may call your data function several times to fill a single reserved location.</li> </ul> |
| <code>refCon</code>    | Contains the reference constant you specified when you assigned your data function to the sequence grabber.  |

**DESCRIPTION**

The sequence grabber calls your data function whenever any channel component writes data to the destination movie. You may use your data function to store the digitized data in some format other than a QuickTime movie.

**RESULT CODES**

File Manager errors  
Memory Manager errors

**SEE ALSO**

You can instruct the sequence grabber not to write its data to a QuickTime movie by calling the `SGSetDataOutput` function and setting the `seqGrabDontMakeMovie` flag to 1. This can save processing time in cases where you do not want to create or update a movie. `SGSetDataOutput` is discussed on page 5-26.

## MyModalFilter

---

The `SGSettingsDialog` function causes the sequence grabber to present its settings dialog box to the user. This is a movable modal dialog box, so you must provide a filter function to handle update events in your window. You specify your filter function with the `proc` parameter.

A modal-dialog filter function whose address is passed to `SGSettingsDialog` should support the following interface:

```
pascal Boolean MyModalFilter (DialogPtr theDialog,
                             EventRecord *theEvent,
                             short *itemHit, long refCon);
```

|                        |   |
|------------------------|---|
| <code>theDialog</code> | Points to the settings dialog box's dialog structure.   |
| <code>theEvent</code>  | Contains a pointer to an event structure. This event structure contains information identifying the nature of the event.  |
| <code>itemHit</code>   | Contains a pointer to a field that contains the item selected by the user. If you handle the event, you should update this field to reflect the item number of the selected item.                                   |
| <code>refCon</code>    | Contains a reference constant. You provide this reference constant to the sequence grabber in the <code>procRefNum</code> parameter of the <code>SGSettingsDialog</code> function, which is described on page 5-48. |

### DESCRIPTION

Your modal-dialog filter function returns a Boolean value that indicates whether you handled the event. Set this value to `true` if you handled the event; otherwise, set it to `false`. If you handle the event, be sure to update the value of the field referred to by the `itemHit` parameter.

### SEE ALSO

See *Inside Macintosh: Files* for a sample modal-dialog filter function.



## Summary of Sequence Grabber Components

---

### C Summary

---

#### Constants

---

```

/* sequence grabber component type */
#define SeqGrabComponentType 'barg'

/* sequence grabber channel type */
#define SeqGrabChannelType 'sgch'

/* SGGGrabPict function grabPictFlags parameter flags */
enum {
    grabPictOffScreen = 1,      /* place in offscreen graphics world */
    grabPictIgnoreClip = 2     /* ignore channel clipping regions */
};

/* flag for SGSetFlags and SGGetFlags functions */
#define sgFlagControlledGrab (1)/* controlled grab */

/* flags for SGSetChannelPlayFlags and SGGetChannelPlayFlags functions */
#define channelPlayNormal 0     /* use default playback methodology */
#define channelPlayFast 1      /* achieve fast playback rate */
#define channelPlayHighQuality 2 /* achieve high quality image */
#define channelPlayAllData 4    /* play all captured data */

/* flags for SGSetDataOutput and SGGetDataOutput functions */
enum {
    seqGrabToDisk          = 1, /* write recorded data to movie */
    seqGrabToMemory        = 2, /* store recorded data in memory */
    seqGrabDontUseTempMemory = 4, /* no temporary memory for recorded
                                   data */
    seqGrabAppendToFile     = 8, /* add recorded data to file's data
                                   fork */
    seqGrabDontAddMovieResource = 16, /* don't add movie resource to file */
    seqGrabDontMakeMovie    = 32 /* don't put data into movie */
};

typedef unsigned char SeqGrabDataOutputEnum;

```

## Sequence Grabber Components

```

/* usage flags for SGSetChannelUsage and SGGetChannelUsage functions */
enum {
    seqGrabRecord          = 1,  /* used during record operations */
    seqGrabPreview          = 2,  /* used during preview operations */
    seqGrabPlayDuringRecord = 4    /* plays data during record operation */
};

typedef unsigned char SeqGrabUsageEnum;

/* SGGetChannelInfo function flags */
enum {
    seqGrabHasBounds          = 1,  /* visual representation of data */
    seqGrabHasVolume          = 2,  /* audio representation of data */
    seqGrabHasDiscreteSamples = 4    /* data organized in discrete frames */
}; typedef unsigned char SeqGrabChannelInfoEnum;

/* device list structure flags */

#define sgDeviceListWithIcons (1)          /* include icons */
#define sgDeviceListDontCheckAvailability (2) /* don't check available */

/* data function write operation types */

enum {
    seqGrabWriteAppend,          /* append to file */
    seqGrabWriteReserve,        /* reserve space in file */
    seqGrabWriteFill             /* fill reserved space */
};

/* SGPause and SGGetPause options */

enum {
    seqGrabUnpause = 0,          /* release grabber */
    seqGrabPause = 1,           /* pause all playback */
    seqGrabPauseForMenu = 3      /* pause for menu display */
};

/* selectors for basic sequence grabber component functions */
kSGInitializeSelect          = 0x1;  /* SGInitialize */
kSGSetDataOutputSelect       = 0x2;  /* SGSetDataOutput */
kSGGetDataOutputSelect       = 0x3;  /* SGGetDataOutput */
kSGSetGWorldSelect           = 0x4;  /* SGSetGWorld */
kSGGetGWorldSelect           = 0x5;  /* SGGetGWorld */
kSGNewChannelSelect          = 0x6;  /* SGNewChannel */

```

## Sequence Grabber Components

```

kSGDisposeChannelSelect      = 0x7;    /* SGDisposeChannel */
kSGStartPreviewSelect        = 0x10;   /* SGStartPreview */
kSGStartRecordSelect         = 0x11;   /* SGStartRecord */
kSGIdleSelect                = 0x12;   /* SGIdle */
kSGStopSelect                = 0x13;   /* SGStop */
kSGPauseSelect               = 0x14;   /* SGPause */
kSGPrepareSelect             = 0x15;   /* SGPrepare */
kSGReleaseSelect             = 0x16;   /* SGRelease */
kSGGetMovieSelect            = 0x17;   /* SGGetMovie */
kSGSetMaximumRecordTimeSelect = 0x18;   /* SGSetMaximumRecordTime */
kSGGetMaximumRecordTimeSelect = 0x19;   /* SGGetMaximumRecordTime */
kSGGetStorageSpaceRemainingSelect = 0x1a; /* SGGetStorageSpaceRemaining */
kSGGetTimeRemainingSelect    = 0x1b;   /* SGGetTimeRemaining */
kSGGrabPictSelect            = 0x1c;   /* SGGrabPict */
kSGGetLastMovieResIDSelect    = 0x1d;   /* SGGetLastMovieResID */
kSGSetFlagsSelect            = 0x1e;   /* SGSetFlags */
kSGGetFlagsSelect            = 0x1f;   /* SGGetFlags */
kSGSetDataProcSelect         = 0x20;   /* SGSetDataProc */
kSGNewChannelFromComponentSelect = 0x21; /* SGNewChannelFromComponent */
kSGDisposeDeviceListSelect    = 0x22;   /* SGDisposeDeviceList */
kSGAppendDeviceListToMenuSelect = 0x23; /* SGAppendDeviceListToMenu */
kSGSetSettingsSelect         = 0x24;   /* SGSetSettings */
kSGGetSettingsSelect         = 0x25;   /* SGGetSettings */
kSGGetIndChannelSelect        = 0x26;   /* SGGetIndChannel */
kSGUpdateSelect              = 0x27;   /* SGUpdate */
kSGGetPauseSelect            = 0x28;   /* SGGetPause */
kSGSettingsDialogSelect       = 0x29;   /* SGSettingsDialog */
kSGGetAlignmentProcSelect     = 0x2A;   /* SGGetAlignmentProc */
kSGSetChannelSettingsSelect    = 0x2B;   /* SGSetChannelSettings */
kSGGetChannelSettingsSelect    = 0x2C;   /* SGGetChannelSettings */

/* selectors for common channel configuration functions */
kSGCSetChannelUsageSelect     = 0x80;   /* SGCSetChannelUsage */
kSGCGetChannelUsageSelect     = 0x81;   /* SGCGetChannelUsage */
kSGCSetChannelBoundsSelect    = 0x82;   /* SGCSetChannelBounds */
kSGCGetChannelBoundsSelect    = 0x83;   /* SGCGetChannelBounds */
kSGCSetChannelVolumeSelect    = 0x84;   /* SGCSetChannelVolume */
kSGCGetChannelVolumeSelect    = 0x85;   /* SGCGetChannelVolume */
kSGCGetChannelInfoSelect      = 0x86;   /* SGCGetChannelInfo */
kSGCSetChannelPlayFlagsSelect = 0x87;   /* SGCSetChannelPlayFlags */
kSGCGetChannelPlayFlagsSelect = 0x88;   /* SGCGetChannelPlayFlags */
kSGCSetChannelMaxFramesSelect = 0x89;   /* SGCSetChannelMaxFrames */
kSGCGetChannelMaxFramesSelect = 0x8a;   /* SGCGetChannelMaxFrames */

```

## Sequence Grabber Components

```

kSGCSetChannelRefConSelect      = 0x8b;      /* SGCSetChannelRefCon */
kSGCSetChannelClipSelect       = 0x8C;      /* SGCSetChannelClip */
kSGCGetChannelClipSelect       = 0x8D;      /* SGCGetChannelClip */
kSGCGetChannelSampleDescriptionSelect = 0x8E;
                                /* SGCGetChannelSampleDescription */
kSGCGetChannelDeviceListSelect = 0x8F;      /* SGCGetChannelDeviceList */
kSGCSetChannelDeviceSelect     = 0x90;      /* SGCSetChannelDevice */
kSGCSetChannelMatrixSelect     = 0x91;      /* SGCSetChannelMatrix */
kSGCGetChannelMatrixSelect     = 0x92;      /* SGCGetChannelMatrix */
kSGCGetChannelTimeScaleSelect  = 0x93;      /* SGCGetChannelTimeScale */

/* selectors for video channel configuration functions */
kSGCGetSrcVideoBoundsSelect     = 0x100;     /* SGCGetSrcVideoBounds */
kSGCSetVideoRectSelect         = 0x101;     /* SGCSetVideoRect */
kSGCGetVideoRectSelect         = 0x102;     /* SGCGetVideoRect */
kSGCGetVideoCompressorTypeSelect = 0x103;   /* SGCGetVideoCompressorType */
kSGCSetVideoCompressorTypeSelect = 0x104;   /* SGCSetVideoCompressorType */
kSGCSetVideoCompressorSelect    = 0x105;    /* SGCSetVideoCompressor */
kSGCGetVideoCompressorSelect    = 0x106;    /* SGCGetVideoCompressor */
kSGCGetVideoDigitizerComponentSelect
                                = 0x107;
                                /* SGCGetVideoDigitizerComponent */
kSGCSetVideoDigitizerComponentSelect
                                = 0x108;
                                /* SGCSetVideoDigitizerComponent */
kSGCVideoDigitizerChangedSelect = 0x109;   /* SGCVideoDigitizerChanged */
kSGCSetVideoBottlenecksSelect   = 0x10a;   /* SGCSetVideoBottlenecks */
kSGCGetVideoBottlenecksSelect   = 0x10b;   /* SGCGetVideoBottlenecks */
kSGCGrabFrameSelect             = 0x10c;   /* SGCGrabFrame */
kSGCGrabFrameCompleteSelect     = 0x10d;   /* SGCGrabFrameComplete */
kSGCDisplayFrameSelect          = 0x10e;   /* SGCDisplayFrame */
kSGCCompressFrameSelect         = 0x10f;   /* SGCCompressFrame */
kSGCCompressFrameCompleteSelect = 0x110;   /* SGCCompressFrameComplete */
kSGCAddFrameSelect              = 0x111;   /* SGCAddFrame */
kSGCTransferFrameForCompressSelect = 0x112;
                                /* SGCTransferFrameForCompress */
kSGCSetCompressBufferSelect     = 0x113;
                                /* SGCSetCompressBuffer */
kSGCGetCompressBufferSelect     = 0x114;
                                /* SGCGetCompressBuffer */
kSGCGetBufferInfoSelect         = 0x115;   /* SGCGetBufferInfo */
kSGCSetUseScreenBufferSelect    = 0x116;   /* SGCSetUseScreenBuffer */
kSGCGetUseScreenBufferSelect    = 0x117;   /* SGCGetUseScreenBuffer */

```

## Sequence Grabber Components

```

kSGCGrabCompressCompleteSelect    = 0x118; /* SGCGrabCompressComplete */
kSGCDisplayCompressSelect         = 0x119; /* SGCDisplayCompress */
kSGCSetFrameRateSelect           = 0x11A; /* SGCSetFrameRate */
kSGCGetFrameRateSelect           = 0x11B; /* SGCGetFrameRate */

/* selectors for sound channel configuration functions */
kSGCSetSoundInputDriverSelect     = 0x100; /* SGCSetSoundInputDriver */
kSGCGetSoundInputDriverSelect     = 0x101; /* SGCGetSoundInputDriver */
kSGCSoundInputDriverChangedSelect = 0x102;
                                   /* SGCSoundInputDriverChanged */
kSGCSetSoundRecordChunkSizeSelect = 0x103;
                                   /* SGCSetSoundRecordChunkSize */
kSGCGetSoundRecordChunkSizeSelect = 0x104;
                                   /* SGCGetSoundRecordChunkSize */
kSGCSetSoundInputRateSelect       = 0x105; /* SGCSetSoundInputRate */
kSGCGetSoundInputRateSelect       = 0x106; /* SGCGetSoundInputRate */
kSGCSetSoundInputParametersSelect = 0x107;
                                   /* SGCSetSoundInputParameters */
kSGCGetSoundInputParametersSelect = 0x108;
                                   /* SGCGetSoundInputParameters */

/* selectors for utility functions provided to channel components */
kSGWriteMovieDataSelect           = 0x100; /* SGWriteMovieData */
kSGAddFrameReferenceSelect        = 0x101; /* SGAddFrameReference */
kSGGetNextFrameReferenceSelect    = 0x102; /* SGGetNextFrameReference */
kSGGetTimeBaseSelect             = 0x103; /* SGGetTimeBase */
kSGSortDeviceListSelect          = 0x104; /* SGSortDeviceList */
kSGAddMovieDataSelect            = 0x105; /* SGAddMovieData */
kSGChangedSourceSelect           = 0x106; /* SGChangedSource */

```

## Data Types

---

```

struct SGCompressInfo {
    Ptr          buffer;      /* buffer for compressed image */
    unsigned long bufferSize; /* bytes of image data in buffer */
    unsigned char similarity; /* relative similarity */
    unsigned char reserved;   /* reserved--set to 0 */
};

typedef struct SGCompressInfo SGCompressInfo;

struct SeqGrabFrameInfo {
    long    frameOffset; /* offset to the sample */
    long    frameTime;   /* time that frame was captured */

```

## Sequence Grabber Components

```

    long      frameSize;      /* number of bytes in sample */
    SGChannel  frameChannel;   /* current connection to channel */
    long      frameRefCon;     /* reference constant for channel */
};

struct VideoBottles {
    short      procCount;      /* count of callbacks */
    GrabProc   grabProc;       /* grab function */
    GrabCompleteProc grabCompleteProc; /* grab-complete
                                   function */
    DisplayProc displayProc;    /* display function */
    CompressProc compressProc;  /* compress function */
    CompressCompleteProc compressCompleteProc; /* compress-complete
                                                function */
    AddFrameProc addFrameProc; /* add-frame function */
    TransferFrameProc transferFrameProc; /* transfer-frame
                                           function */
    GrabCompressCompleteProc grabCompressCompleteProc; /* grab-compress-complete
                                                         function */
    DisplayCompressProc displayCompressProc; /* display-compress
                                              function */
};

typedef struct VideoBottles VideoBottles;

typedef struct SGDeviceListRecord {
    short      count;          /* count of devices */
    short      selectedIndex;  /* current device */
    long      reserved;        /* set to 0 */
    SGDeviceName entry[1];     /* device names */
} SGDeviceListRecord, *SGDeviceListPtr, **SGDeviceList;

typedef struct SGDeviceName {
    Str63      name;           /* device name */
    Handle     icon;           /* device icon */
    long      flags;           /* flags */
    long      refCon;          /* set to 0 */
    long      reserved;        /* set to 0 */
} SGDeviceName;

```

## Sequence Grabber Component Functions

---

### Configuring Sequence Grabber Components

```

pascal ComponentResult SGInitialize
    (SeqGrabComponent s);

pascal ComponentResult SGSetDataOutput
    (SeqGrabComponent s, FSSpec *movieFile,
     long whereFlags);

pascal ComponentResult SGGetDataOutput
    (SeqGrabComponent s,
     FSSpec *movieFile, long *whereFlags);

pascal ComponentResult SGSetGWorld
    (SeqGrabComponent s, CGrafPtr gp, GDHandle gd);

pascal ComponentResult SGGetGWorld
    (SeqGrabComponent s, CGrafPtr *gp,
     GDHandle *gd);

pascal ComponentResult SGNewChannel
    (SeqGrabComponent s, OSType channelType,
     SGChannel *ref);

pascal ComponentResult SGNewChannelFromComponent
    (SeqGrabComponent s, SGChannel *newChannel,
     Component sgChannelComponent);

pascal ComponentResult SGGetIndChannel
    (SeqGrabComponent s, short index,
     SGChannel *ref, OSType *chanType);

pascal ComponentResult SGDisposeChannel
    (SeqGrabComponent s, SGChannel c);

pascal ComponentResult SGSetDataProc
    (SeqGrabComponent sg, SGDataProc proc,
     long refCon);

pascal ComponentResult SGGetAlignmentProc
    (SeqGrabComponent s,
     AlignmentProcRecordPtr alignmentProc);

```

### Controlling Sequence Grabber Components

```

pascal ComponentResult SGStartPreview
    (SeqGrabComponent s);

pascal ComponentResult SGStartRecord
    (SeqGrabComponent s);

pascal ComponentResult SGIdle
    (SeqGrabComponent s);

```

## Sequence Grabber Components

```

pascal ComponentResult SGUpdate
    (SeqGrabComponent s, RgnHandle updateRgn);
pascal ComponentResult SGStop
    (SeqGrabComponent s);
pascal ComponentResult SGPause
    (SeqGrabComponent s, Byte pause);
pascal ComponentResult SGGetPause
    (SeqGrabComponent s, Byte *paused);
pascal ComponentResult SGPrepare
    (SeqGrabComponent s, Boolean prepareForPreview,
     Boolean prepareForRecord);
pascal ComponentResult SGRelease
    (SeqGrabComponent s);
pascal Movie SGGetMovie    (SeqGrabComponent s);
pascal ComponentResult SGGetLastMovieResID
    (SeqGrabComponent s, short *resID);
pascal ComponentResult SGGrabPict
    (SeqGrabComponent s, PicHandle *p,
     const Rect *bounds, short offscreenDepth,
     long grabPictFlags);

```

**Working With Sequence Grabber Settings**

```

pascal ComponentResult SGSettingsDialog
    (SeqGrabComponent s, SGChannel c,
     short numPanels, Component *panelList,
     long flags, SGModalFilterProcPtr proc,
     long procRefNum);
pascal ComponentResult SGGetSettings
    (SeqGrabComponent s, UserData *ud, long flags);
pascal ComponentResult SGSetSettings
    (SeqGrabComponent s, UserData ud, long flags);
pascal ComponentResult SGGetChannelSettings
    (SeqGrabComponent s, SGChannel c, UserData *ud,
     long flags);
pascal ComponentResult SGSetChannelSettings
    (SeqGrabComponent s, SGChannel c, UserData ud,
     long flags);

```

**Working With Sequence Grabber Characteristics**

```

pascal ComponentResult SGSetMaximumRecordTime
    (SeqGrabComponent s, unsigned long ticks);

```



## Sequence Grabber Components

```

pascal ComponentResult SGGetMaximumRecordTime
    (SeqGrabComponent s, unsigned long *ticks);
pascal ComponentResult SGGetStorageSpaceRemaining
    (SeqGrabComponent s, unsigned long *bytes);
pascal ComponentResult SGGetTimeRemaining
    (SeqGrabComponent s, long *ticksLeft);
pascal ComponentResult SGGetTimeBase
    (SeqGrabComponent s, TimeBase *tb);
pascal ComponentResult SGSetFlags
    (SeqGrabComponent s, long sgFlags);
pascal ComponentResult SGGetFlags
    (SeqGrabComponent s, long *sgFlags);

```

**Working With Channel Characteristics**

```

pascal ComponentResult SGSetChannelUsage
    (SGChannel c, long usage);
pascal ComponentResult SGGetChannelUsage
    (SGChannel c, long *usage);
pascal ComponentResult SGGetChannelInfo
    (SGChannel c, long *channelInfo);
pascal ComponentResult SGSetChannelPlayFlags
    (SGChannel c, long playFlags);
pascal ComponentResult SGGetChannelPlayFlags
    (SGChannel c, long *playFlags);
pascal ComponentResult SGSetChannelMaxFrames
    (SGChannel c, long frameCount);
pascal ComponentResult SGGetChannelMaxFrames
    (SGChannel c, long *frameCount);
pascal ComponentResult SGSetChannelBounds
    (SGChannel c, const Rect *bounds);
pascal ComponentResult SGGetChannelBounds
    (SGChannel c, Rect *bounds);
pascal ComponentResult SGSetChannelVolume
    (SGChannel c, short volume);
pascal ComponentResult SGGetChannelVolume
    (SGChannel c, short *volume);
pascal ComponentResult SGSetChannelRefCon
    (SGChannel c, long refCon);
pascal ComponentResult SGGetChannelSampleDescription
    (SGChannel c, Handle sampleDesc);
pascal ComponentResult SGGetChannelTimeScale
    (SGChannel c, TimeScale *scale);

```

## Sequence Grabber Components

```

pascal ComponentResult SGSetChannelClip
    (SGChannel c, RgnHandle theClip);
pascal ComponentResult SGGetChannelClip
    (SGChannel c, RgnHandle *theClip);
pascal ComponentResult SGSetChannelMatrix
    (SGChannel c, const MatrixRecord *m);
pascal ComponentResult SGGetChannelMatrix
    (SGChannel c, MatrixRecord *m);

```

**Working With Channel Devices**

```

pascal ComponentResult SGGetChannelDeviceList
    (SGChannel c, long selectionFlags,
     SGDeviceList *list);
pascal ComponentResult SGDisposeDeviceList
    (SeqGrabComponent s, SGDeviceList list);
pascal ComponentResult SGAppendDeviceListToMenu
    (SeqGrabComponent s, SGDeviceList list,
     MenuHandle mh);
pascal ComponentResult SGSetChannelDevice
    (SGChannel c, StringPtr name);

```

**Working With Video Channels**

```

pascal ComponentResult SGGetSrcVideoBounds
    (SGChannel c, Rect *r);
pascal ComponentResult SGSetVideoRect
    (SGChannel c, Rect *r);
pascal ComponentResult SGGetVideoRect
    (SGChannel c, Rect *r);
pascal ComponentResult SGSetVideoCompressorType
    (SGChannel c, OSType compressorType);
pascal ComponentResult SGGetVideoCompressorType
    (SGChannel c, OSType *compressorType);
pascal ComponentResult SGSetVideoCompressor
    (SGChannel c, short depth,
     CompressorComponent compressor,
     CodecQ spatialQuality,
     CodecQ temporalQuality, long keyFrameRate);
pascal ComponentResult SGGetVideoCompressor
    (SGChannel c, short *depth,
     CompressorComponent *compressor,
     CodecQ *spatialQuality,
     CodecQ *temporalQuality, long *keyFrameRate);

```

## Sequence Grabber Components

```

pascal ComponentResult SGSetVideoDigitizerComponent
    (SGChannel c, ComponentInstance vdig);
pascal ComponentInstance SGGetVideoDigitizerComponent
    (SGChannel c);
pascal ComponentResult SGVideoDigitizerChanged
    (SGChannel c);
pascal ComponentResult SGSetCompressBuffer
    (SGChannel c, short depth,
     const Rect *compressSize);
pascal ComponentResult SGGetCompressBuffer
    (SGChannel c, short *depth, Rect *compressSize);
pascal ComponentResult SGSetFrameRate
    (SGChannel c, Fixed frameRate);
pascal ComponentResult SGGetFrameRate
    (SGChannel c, Fixed *frameRate);
pascal ComponentResult SGSetUseScreenBuffer
    (SGChannel c, Boolean useScreenBuffer);
pascal ComponentResult SGGetUseScreenBuffer
    (SGChannel c, Boolean *useScreenBuffer);

```

**Working With Sound Channels**

```

pascal ComponentResult SGSetSoundInputDriver
    (SGChannel c, const Str255 driverName);
pascal long SGGetSoundInputDriver
    (SGChannel c);
pascal ComponentResult SGSoundInputDriverChanged
    (SGChannel c);
pascal ComponentResult SGSetSoundRecordChunkSize
    (SGChannel c, long seconds);
pascal long SGGetSoundRecordChunkSize
    (SGChannel c);
pascal ComponentResult SGSetSoundInputRate
    (SGChannel c, Fixed rate);
pascal Fixed SGGetSoundInputRate
    (SGChannel c);
pascal ComponentResult SGSetSoundInputParameters
    (SGChannel c, short sampleSize,
     short numChannels, OSType compressionType);
pascal ComponentResult SGGetSoundInputParameters
    (SGChannel c, short *sampleSize,
     short *numChannels, OSType *compressionType);

```

## Sequence Grabber Components

**Video Channel Callback Functions**

```
pascal ComponentResult SGSetVideoBottlenecks
    (SGChannel c, VideoBottles *vb);

pascal ComponentResult SGGetVideoBottlenecks
    (SGChannel c, VideoBottles *vb);
```

**Utility Functions for Video Channel Callback Functions**

```
pascal ComponentResult SGGetBufferInfo
    (SGChannel c, short bufferNum,
     PixMapHandle *bufferPM, Rect *bufferRect,
     GWorldPtr *compressBuffer,
     Rect *compressBufferRect);

pascal ComponentResult SGGrabFrame
    (SGChannel c, short bufferNum);

pascal ComponentResult SGGrabFrameComplete
    (SGChannel c, short bufferNum, Boolean *done);

pascal ComponentResult SGDisplayFrame
    (SGChannel c, short bufferNum,
     MatrixRecord *mp, RgnHandle clipRgn);

pascal ComponentResult SGCompressFrame
    (SGChannel c, short bufferNum);

pascal ComponentResult SGCompressFrameComplete
    (SGChannel c, short bufferNum, Boolean *done,
     SGCompressInfo *ci);

pascal ComponentResult SGAddFrame
    (SGChannel c, short bufferNum,
     TimeValue atTime, TimeScale scale,
     const SGCompressInfo *ci);

pascal ComponentResult SGTransferFrameForCompress
    (SGChannel c, short bufferNum, MatrixRecord *mp,
     RgnHandle clipRgn);

pascal ComponentResult SGGrabCompressComplete
    (SGChannel c, Boolean *done,
     SGCompressInfo *ci, TimeRecord *tr);

pascal ComponentResult SGDisplayCompress
    (SGChannel c, Ptr dataPtr,
     ImageDescriptionHandle desc, MatrixRecord *mp,
     RgnHandle clipRgn);
```

## Application-Defined Functions

---

```

pascal ComponentResult MyGrabFunction
    (SGChannel c, short bufferNum, long refCon);

pascal ComponentResult MyGrabCompleteFunction
    (SGChannel c, short bufferNum, Boolean *done,
     long refCon);

pascal ComponentResult MyDisplayFunction
    (SGChannel c, short bufferNum,
     MatrixRecord *mp, RgnHandle clipRgn,
     long refCon);

pascal ComponentResult MyCompressFunction
    (SGChannel c, short bufferNum, long refCon);

pascal ComponentResult MyCompressCompleteFunction
    (SGChannel c, short bufferNum, Boolean *done,
     SGCompressInfo *ci, long refCon);

pascal ComponentResult MyAddFrameFunction
    (SGChannel c, short bufferNum,
     TimeValue atTime, TimeScale scale,
     SGCompressInfo ci, long refCon);

pascal ComponentResult MyTransferFrameFunction
    (SGChannel c, short bufferNum, MatrixRecord *mp,
     RgnHandle clipRgn, long refCon);

pascal ComponentResult MyGrabCompressCompleteFunction
    (SGChannel c, Boolean *done,
     SGCompressInfo *ci, TimeRecord *tr,
     long refCon);

pascal ComponentResult MyDisplayCompressFunction
    (SGChannel c, Ptr dataPtr,
     ImageDescriptionHandle desc, MatrixRecord *mp,
     RgnHandle clipRgn, long refCon);

pascal OSErr MyDataFunction (SGChannel c, Ptr p, long len, long *offset,
                             long chRefCon, TimeValue time,
                             short writeType, long refCon);

pascal Boolean MyModalFilter
    (DialogPtr theDialog, EventRecord *theEvent,
     short *itemHit, long refCon);

```

## Pascal Summary

---

### Constants

---

CONST

```

{sequence grabber component type}
SeqGrabComponentType      = 'barg';

{sequence grabber channel type}
SeqGrabChannelType        = 'sgch'

{SGGrabPict function grabPictFlags parameter flags}
grabPictOffScreen          = 1;  {place in offscreen graphics world}
grabPictIgnoreClip         = 2;  {ignore channel clipping regions}

{flag for SGSetFlags and SGGetFlags functions}
sgFlagControlledGrab       = 1;  {controlled grab}

{flags for SGSetChannelPlayFlags and SGGetChannelPlayFlags functions}
channelPlayNormal          = 0;  {use default playback methodology}
channelPlayFast            = 1;  {achieve fast playback rate}
channelPlayHighQuality     = 2;  {achieve high quality image}
channelPlayAllData         = 4;  {play all captured data}

{flags for SGSetDataOutput and SGGetDataOutput functions}
seqGrabToDisk              = 1;  {write recorded data to specified }
                               { QuickTime movie}
seqGrabToMemory            = 2;  {store recorded data in memory until }
                               { completion of recording process}
seqGrabDontUseTempMemory   = 4;  {don't use temporary memory to store }
                               { recorded data}
seqGrabAppendToFile        = 8;  {add recorded data to data fork of }
                               { specified movie file}
seqGrabDontAddMovieResource = 16; {don't add movie resource to }
                               { specified movie file}

{usage flags for SGSetChannelUsage and SGGetChannelUsage functions}
seqGrabRecord              = 1;  {used during record operations}
seqGrabPreview             = 2;  {used during preview operations}
seqGrabPlayDuringRecord    = 4;  {used during record operations}

```

## Sequence Grabber Components

```

{SGGetChannelInfo function flags}
seqGrabHasBounds           = 1;  {visual representation of data}
seqGrabHasVolume           = 2;  {audio representation of data}
seqGrabHasDiscreteSamples  = 4;  {data organized in discrete frames}

{device list structure flags}
sgDeviceListWithIcons      = 1;  {include icons}
sgDeviceListDontCheckAvailability = 2;  {do not check availability }
                                   { of device}

{data function write operation types}
seqGrabWriteAppend         = 0;  {append to file}
seqGrabWriteReserve        = 1;  {reserve space in file}
seqGrabWrite               = 2;  {fill reserved space}

{SGPause and SGGetPause options}
seqGrabUnpause            = 0;    {release grabber}
seqGrabPause              = 1;    {pause all playback}
seqGrabPauseForMenu       = 3;    {pause for menu display}

{selectors for basic sequence grabber component functions}
kSGInitializeSelect       = $1;    {SGInitialize}
kSGSetDataOutputSelect    = $2;    {SGSetDataOutput}
kSGGetDataOutputSelect    = $3;    {SGGetDataOutput}
kSGSetGWorldSelect        = $4;    {SGSetGWorld}
kSGGetGWorldSelect        = $5;    {SGGetGWorld}
kSGNewChannelSelect       = $6;    {SGNewChannel}
kSGDisposeChannelSelect   = $7;    {SGDisposeChannel}
kSGStartPreviewSelect     = $10;   {SGStartPreview}
kSGStartRecordSelect      = $11;   {SGStartRecord}
kSGIdleSelect             = $12;   {SGIdle}
kSGStopSelect             = $13;   {SGStop}
kSGPauseSelect            = $14;   {SGPause}
kSGPrepareSelect          = $15;   {SGPrepare}
kSGReleaseSelect          = $16;   {SGRelease}
kSGGetMovieSelect         = $17;   {SGGetMovie}
kSGSetMaximumRecordTimeSelect = $18; {SGSetMaximumRecordTime}
kSGGetMaximumRecordTimeSelect = $19; {SGGetMaximumRecordTime}
kSGGetStorageSpaceRemainingSelect = $1A; {SGGetStorageSpaceRemaining}
kSGGetTimeRemainingSelect = $1B;   {SGGetTimeRemaining}
kSGGrabPictSelect         = $1C;   {SGGrabPict}
kSGGetLastMovieResIDSelect = $1D;   {SGGetLastMovieResID}
kSGSetFlagsSelect         = $1E;   {SGSetFlags}
kSGGetFlagsSelect         = $1F;   {SGGetFlags}

```

## Sequence Grabber Components

```

kSGSetDataProcSelect          = $20;    {SGSetDataProc}
kSGNewChannelFromComponentSelect = $21;    {SGNewChannelFromComponent}
kSGDisposeDeviceListSelect     = $22;    {SGDisposeDeviceList}
kSGAppendDeviceListToMenuSelect = $23;    {SGAppendDeviceListToMenu}
kSGSetSettingsSelect          = $24;    {SGSetSettings}
kSGGetSettingsSelect          = $25;    {SGGetSettings}
kSGGetIndChannelSelect        = $26;    {SGGetIndChannel}
kSGUpdateSelect               = $27;    {SGUpdate}
kSGGetPauseSelect             = $28;    {SGGetPause}
kSGSettingsDialogSelect       = $29;    {SGSettingsDialog}
kSGGetAlignmentProcSelect     = $2A;    {SGGetAlignmentProc}
kSGSetChannelSettingsSelect    = $2B;    {SGSetChannelSettings}
kSGGetChannelSettingsSelect    = $2C;    {SGGetChannelSettings}

{selectors for common channel configuration functions}
kSGCSetChannelUsageSelect      = $80;    {SGCSetChannelUsage}
kSGCGetChannelUsageSelect      = $81;    {SGCGetChannelUsage}
kSGCSetChannelBoundsSelect     = $82;    {SGCSetChannelBounds}
kSGCGetChannelBoundsSelect     = $83;    {SGCGetChannelBounds}
kSGCSetChannelVolumeSelect     = $84;    {SGCSetChannelVolume}
kSGCGetChannelVolumeSelect     = $85;    {SGCGetChannelVolume}
kSGCGetChannelInfoSelect       = $86;    {SGCGetChannelInfo}
kSGCSetChannelPlayFlagsSelect  = $87;    {SGCSetChannelPlayFlags}
kSGCGetChannelPlayFlagsSelect  = $88;    {SGCGetChannelPlayFlags}
kSGCSetChannelMaxFramesSelect   = $89;    {SGCSetChannelMaxFrames}
kSGCGetChannelMaxFramesSelect   = $8A;    {SGCGetChannelMaxFrames}
kSGCSetChannelRefConSelect     = $8B;    {SGCSetChannelRefCon}
kSGCSetChannelClipSelect       = $8C;    {SGCSetChannelClip}
kSGCGetChannelClipSelect       = $8D;    {SGCGetChannelClip}
kSGCGetChannelSampleDescriptionSelect = $8E;    {SGCGetChannelSampleDescription}
kSGCGetChannelDeviceListSelect  = $8F;    {SGCGetChannelDeviceList}
kSGCSetChannelDeviceSelect      = $90;    {SGCSetChannelDevice}
kSGCSetChannelMatrixSelect      = $91;    {SGCSetChannelMatrix}
kSGCGetChannelMatrixSelect      = $92;    {SGCGetChannelMatrix}
kSGCGetChannelTimeScaleSelect   = $93;    {SGCGetChannelTimeScale}

{selectors for video channel configuration functions}
kSGCGetSrcVideoBoundsSelect     = $100;   {SGCGetSrcVideoBounds}
kSGCSetVideoRectSelect          = $101;   {SGCSetVideoRect}
kSGCGetVideoRectSelect          = $102;   {SGCGetVideoRect}
kSGCGetVideoCompressorTypeSelect = $103;   {SGCGetVideoCompressorType}

```



## Sequence Grabber Components

```

kSGCSetVideoCompressorTypeSelect      = $104;
                                        {SGCSetVideoCompressorType}
kSGCSetVideoCompressorSelect          = $105;  {SGCSetVideoCompressor}
kSGCGetVideoCompressorSelect          = $106;  {SGCGetVideoCompressor}
kSGCGetVideoDigitizerComponentSelect  = $107;
                                        {SGCGetVideoDigitizerComponent}
kSGCSetVideoDigitizerComponentSelect  = $108;
                                        {SGCSetVideoDigitizerComponent}
kSGCVideoDigitizerChangedSelect       = $109;  {SGCVideoDigitizerChanged}
kSGCSetVideoBottlenecksSelect         = $10A;  {SGCSetVideoBottlenecks}
kSGCGetVideoBottlenecksSelect         = $10B;  {SGCGetVideoBottlenecks}
kSGCGrabFrameSelect                  = $10C;  {SGCGrabFrame}
kSGCGrabFrameCompleteSelect           = $10D;  {SGCGrabFrameComplete}
kSGCDisplayFrameSelect                = $10E;  {SGCDisplayFrame}
kSGCCompressFrameSelect               = $10F;  {SGCCompressFrame}
kSGCCompressFrameCompleteSelect       = $110;  {SGCCompressFrameComplete}
kSGCAddFrameSelect                   = $111;  {SGCAddFrame}
kSGCTransferFrameForCompressSelect     = $112;
                                        {SGCTransferFrameForCompress}
kSGCSetCompressBufferSelect           = $113;  {SGCSetCompressBuffer}
kSGCGetCompressBufferSelect           = $114;  {SGCGetCompressBuffer}
kSGCGetBufferInfoSelect               = $115;  {SGCGetBufferInfo}
kSGCSetUseScreenBufferSelect          = $116;  {SGCSetUseScreenBuffer}
kSGCGetUseScreenBufferSelect          = $117;  {SGCGetUseScreenBuffer}
kSGCGrabCompressCompleteSelect        = $118;  {SGCGrabCompressComplete}
kSGCDisplayCompressSelect              = $119;  {SGCDisplayCompress}
kSGCSetFrameRateSelect                = $11A;  {SGCSetFrameRate}
kSGCGetFrameRateSelect                = $11B;  {SGCGetFrameRate}

{selectors for sound channel configuration functions}
kSGCSetSoundInputDriverSelect         = $100;  {SGCSetSoundInputDriver}
kSGCGetSoundInputDriverSelect         = $101;  {SGCGetSoundInputDriver}
kSGCSoundInputDriverChangedSelect     = $102;  {SGCSoundInputDriverChanged}
kSGCSetSoundRecordChunkSizeSelect     = $103;  {SGCSetSoundRecordChunkSize}
kSGCGetSoundRecordChunkSizeSelect     = $104;  {SGCGetSoundRecordChunkSize}
kSGCSetSoundInputRateSelect           = $105;  {SGCSetSoundInputRate}
kSGCGetSoundInputRateSelect           = $106;  {SGCGetSoundInputRate}
kSGCSetSoundInputParametersSelect     = $107;  {SGCSetSoundInputParameters}
kSGCGetSoundInputParametersSelect     = $108;  {SGCGetSoundInputParameters}

{selectors for utility functions provided to channel components}
kSGWriteMovieDataSelect               = $100;  {SGWriteMovieData}
kSGAddFrameReferenceSelect             = $101;  {SGAddFrameReference}
kSGGetNextFrameReferenceSelect         = $102;  {SGGetNextFrameReference}

```

## Sequence Grabber Components

|                         |          |                    |
|-------------------------|----------|--------------------|
| kSGGetTimeBaseSelect    | = \$103; | {SGGetTimeBase}    |
| kSGSortDeviceListSelect | = \$104; | {SGSortDeviceList} |
| kSGAddMovieDataSelect   | = \$105; | {SGAddMovieData}   |
| kSGChangedSourceSelect  | = \$106; | {SGChangedSource}  |

Data Types

---

```

TYPE SGCompressInfo =
    PACKED RECORD
        buffer:      Ptr;      {buffer containing compressed image}
        bufferSize:  LongInt;   {bytes of image data in buffer}
        similarity:   Char;     {relative similarity of image }
                                { to previous image in sequence}
        reserved:    Char;     {reserved}
    END;

VideoBottles =
    RECORD
        procCount:      Integer;      {number of callback }
                                    { routines in record}
        grabProc:        GrabProc;     {grab function}
        grabCompleteProc: GrabCompleteProc; {grab-complete function}
        displayProc:     DisplayProc;  {display function}
        compressProc:     CompressProc; {compress function}
        compressCompleteProc: CompressCompleteProc;
                                    {compress-complete }
                                    { function}
        addFrameProc:    AddFrameProc; {add-frame function}
        transferFrameProc: TransferFrameProc; {transfer-frame }
                                    { function}
    END;

SeqGrabFrameInfo =
    RECORD
        frameOffset: LongInt;      {offset to the sample}
        frameTime:   LongInt;      {time that frame was captured}
        frameSize:   LongInt;      {number of bytes in sample}
        frameChannel: SGChannel;    {current connection to channel}
        frameRefCon: LongInt;      {reference constant for channel}
    END;

```

## Sequence Grabber Components

```

SGDeviceName =
RECORD
    name:    Str63;        {device name}
    icon:    Handle;        {device icon}
    flags:    LongInt;      {flags}
    refCon:   LongInt;      {set to 0}
    reserved: LongInt;      {reserved--set to 0}
END;

SGDeviceListPtr = ^SGDeviceListRecord;
SGDeviceList = ^SGDeviceListPtr;

SGDeviceListRecord =
RECORD
    count:      Integer;          {count of devices}
    selectedIndex: Integer;        {current device}
    reserved:    LongInt;          {reserved--set to 0}
    entry:       ARRAY[0..0] OF SGDeviceName; {device names}
END;

```

## Sequence Grabber Component Routines

**Configuring Sequence Grabber Components**

```

FUNCTION SGInitialize      (s: SeqGrabComponent): ComponentResult;
FUNCTION SGSetDataOutput   (s: SeqGrabComponent; movieFile: FSSpec;
    whereFlags: LongInt): ComponentResult;
FUNCTION SGGetDataOutput   (s: SeqGrabComponent; VAR movieFile: FSSpec;
    VAR whereFlags: LongInt): ComponentResult;
FUNCTION SGSetGWorld       (s: SeqGrabComponent; gp: CGrafPtr;
    gd: GDHandle): ComponentResult;
FUNCTION SGGetGWorld       (s: SeqGrabComponent; VAR gp: CGrafPtr;
    VAR gd: GDHandle): ComponentResult;
FUNCTION SGNewChannel       (s: SeqGrabComponent; channelType: OSType;
    VAR ref: SGChannel): ComponentResult;
FUNCTION SGNewChannelFromComponent
    (s: SeqGrabComponent;
    VAR newChannel: SGChannel;
    sgChannelComponent: Component):
    ComponentResult;
FUNCTION SGGetIndChannel    (s: SeqGrabComponent; index: Integer;
    VAR ref: SGChannel;
    VAR chanType: OSType): ComponentResult;

```

## Sequence Grabber Components

```

FUNCTION SGDisposeChannel    (s: SeqGrabComponent;
                             c: SGChannel): ComponentResult;

FUNCTION SGSetDataProc      (s: SeqGrabComponent; proc: SGDataProc;
                             refCon: LongInt): ComponentResult;

FUNCTION SGGetAlignmentProc (s: SeqGrabComponent;
                             alignmentProc: AlignmentProcRecordPtr):
                             ComponentResult;

```

**Controlling Sequence Grabber Components**

```

FUNCTION SGStartPreview     (s: SeqGrabComponent): ComponentResult;
FUNCTION SGStartRecord      (s: SeqGrabComponent): ComponentResult;
FUNCTION SGIdle             (s: SeqGrabComponent): ComponentResult;
FUNCTION SGUpdate           (s: SeqGrabComponent; updateRgn: RgnHandle):
                             ComponentResult;

FUNCTION SGStop            (s: SeqGrabComponent): ComponentResult;
FUNCTION SGPause           (s: SeqGrabComponent;
                             paused: Byte): ComponentResult;

FUNCTION SGGetPause        (s: SeqGrabComponent;
                             VAR paused: Byte): ComponentResult;

FUNCTION SGPrepare         (s: SeqGrabComponent;
                             prepareForPreview: Boolean;
                             prepareForRecord: Boolean): ComponentResult;

FUNCTION SGRelease         (s: SeqGrabComponent): ComponentResult;
FUNCTION SGGetMovie        (s: SeqGrabComponent): Movie;
FUNCTION SGGetLastMovieResID (s: SeqGrabComponent;
                             VAR resID: Integer): ComponentResult;

FUNCTION SGGrabPict        (s: SeqGrabComponent; VAR p: PicHandle;
                             bounds: Rect; offscreenDepth: Integer;
                             grabPictFlags: LongInt): ComponentResult;

```

**Working With Sequence Grabber Settings**

```

FUNCTION SGSettingsDialog   (s: SeqGrabComponent; c: SGChannel;
                             numPanels: Integer; VAR panelList: Component;
                             flags: LongInt; proc: SGModalFilterProcPtr;
                             procRefNum: LongInt): ComponentResult;

FUNCTION SGGetSettings      (s: SeqGrabComponent; VAR ud: UserData;
                             flags: LongInt): ComponentResult;

FUNCTION SGSetSettings      (s: SeqGrabComponent; ud: UserData;
                             flags: LongInt): ComponentResult;

```

## Sequence Grabber Components

```

FUNCTION SGGetChannelSettings
    (s: SeqGrabComponent; c: SGChannel;
     VAR ud: UserData; flags: LongInt):
     ComponentResult;

FUNCTION SGSetChannelSettings
    (s: SeqGrabComponent; c: SGChannel;
     ud: UserData; flags: LongInt): ComponentResult;

```

**Working With Sequence Grabber Characteristics**

```

FUNCTION SGSetMaximumRecordTime
    (s: SeqGrabComponent; ticks: LongInt):
     ComponentResult;

FUNCTION SGGetMaximumRecordTime
    (s: SeqGrabComponent; VAR ticks: LongInt):
     ComponentResult;

FUNCTION SGGetStorageSpaceRemaining
    (s: SeqGrabComponent; VAR bytes: LongInt):
     ComponentResult;

FUNCTION SGGetTimeRemaining (s: SeqGrabComponent; VAR ticksLeft: LongInt):
     ComponentResult;

FUNCTION SGGetTimeBase      (s: SeqGrabComponent; VAR tb: TimeBase):
     ComponentResult;

FUNCTION SGSetFlags          (s: SeqGrabComponent; sgFlags: LongInt):
     ComponentResult;

FUNCTION SGGetFlags          (s: SeqGrabComponent; VAR sgFlags: LongInt):
     ComponentResult;

```

**Working With Channel Characteristics**

```

FUNCTION SGSetChannelUsage (c: SGChannel; usage: LongInt): ComponentResult;
FUNCTION SGGetChannelUsage (c: SGChannel; VAR usage: LongInt):
     ComponentResult;

FUNCTION SGGetChannelInfo  (c: SGChannel; VAR channelInfo: LongInt):
     ComponentResult;

FUNCTION SGSetChannelPlayFlags
    (c: SGChannel; playFlags: LongInt):
     ComponentResult;

FUNCTION SGGetChannelPlayFlags
    (c: SGChannel; VAR playFlags: LongInt):
     ComponentResult;

FUNCTION SGSetChannelMaxFrames
    (c: SGChannel; frameCount: LongInt):
     ComponentResult;

```

## Sequence Grabber Components

```

FUNCTION SGGetChannelMaxFrames
    (c: SGChannel; VAR frameCount: LongInt):
        ComponentResult;

FUNCTION SGSetChannelBounds
    (c: SGChannel; bounds: Rect): ComponentResult;

FUNCTION SGGetChannelBounds
    (c: SGChannel; VAR bounds: Rect):
        ComponentResult;

FUNCTION SGSetChannelVolume
    (c: SGChannel; volume: Integer):
        ComponentResult;

FUNCTION SGGetChannelVolume
    (c: SGChannel; VAR volume: Integer):
        ComponentResult;

FUNCTION SGSetChannelRefCon
    (c: SGChannel; refCon: LongInt):
        ComponentResult;

FUNCTION SGGetChannelSampleDescription
    (c: SGChannel; sampleDesc: Handle):
        ComponentResult;

FUNCTION SGGetChannelTimeScale
    (c: SGChannel; VAR scale: TimeScale):
        ComponentResult;

FUNCTION SGGetChannelClip
    (c: SGChannel; VAR theClip: RgnHandle):
        ComponentResult;

FUNCTION SGGetChannelClip
    (c: SGChannel; VAR theClip: RgnHandle):
        ComponentResult;

FUNCTION SGGetChannelMatrix
    (c: SGChannel; VAR m: MatrixRecord):
        ComponentResult;

FUNCTION SGGetChannelMatrix
    (c: SGChannel; VAR m: MatrixRecord):
        ComponentResult;

```

**Working With Channel Devices**

```

FUNCTION SGGetChannelDeviceList
    (c: SGChannel; selectionFlags: LongInt;
     VAR list: SGDeviceList): ComponentResult;

FUNCTION SGDisposeDeviceList
    (s: SeqGrabComponent; list: SGDeviceList):
        ComponentResult;

```

## Sequence Grabber Components

```

FUNCTION SGAppendDeviceListToMenu
    (s: SeqGrabComponent; list: SGDeviceList;
     mh: MenuHandle): ComponentResult;

FUNCTION SGSetChannelDevice (c: SGChannel; name: StringPtr):
    ComponentResult;

```

**Working With Video Channels**

```

FUNCTION SGGetSrcVideoBounds
    (c: SGChannel; VAR r: Rect): ComponentResult;

FUNCTION SGSetVideoRect    (c: SGChannel; r: Rect): ComponentResult;
FUNCTION SGGetVideoRect    (c: SGChannel; VAR r: Rect): ComponentResult;
FUNCTION SGSetVideoCompressorType
    (c: SGChannel; compressorType: OSType):
    ComponentResult;

FUNCTION SGGetVideoCompressorType
    (c: SGChannel; VAR compressorType: OSType):
    ComponentResult;

FUNCTION SGSetVideoCompressor
    (c: SGChannel; depth: Integer;
     compressor: CompressorComponent;
     spatialQuality: CodecQ;
     temporalQuality: CodecQ;
     keyFrameRate: LongInt): ComponentResult;

FUNCTION SGGetVideoCompressor
    (c: SGChannel; VAR depth: Integer;
     VAR compressor: CompressorComponent;
     VAR spatialQuality: CodecQ;
     VAR temporalQuality: CodecQ;
     VAR keyFrameRate: LongInt): ComponentResult;

FUNCTION SGSetVideoDigitizerComponent
    (c: SGChannel; vdig: ComponentInstance):
    ComponentResult;

FUNCTION SGGetVideoDigitizerComponent
    (c: SGChannel): ComponentInstance;

FUNCTION SGVideoDigitizerChanged
    (c: SGChannel): ComponentResult;

FUNCTION SGSetCompressBuffer
    (c: SGChannel; depth: Integer;
     compressSize: Rect): ComponentResult;

FUNCTION SGGetCompressBuffer
    (c: SGChannel; VAR depth: Integer;
     VAR compressSize: Rect): ComponentResult;

```

## Sequence Grabber Components

```

FUNCTION SGSetFrameRate      (c: SGChannel;
                             frameRate: Fixed): ComponentResult;

FUNCTION SGGetFrameRate      (c: SGChannel;
                             VAR frameRate: Fixed): ComponentResult;

FUNCTION SGSetUseScreenBuffer
                             (c: SGChannel; useScreenBuffer: Boolean):
                             ComponentResult;

FUNCTION SGGetUseScreenBuffer
                             (c: SGChannel; VAR useScreenBuffer: Boolean):
                             ComponentResult;

```

**Working With Sound Channels**

```

FUNCTION SGSetSoundInputDriver
                             (c: SGChannel; driverName: Str255):
                             ComponentResult;

FUNCTION SGGetSoundInputDriver
                             (c: SGChannel): LongInt;

FUNCTION SGSoundInputDriverChanged
                             (c: SGChannel): ComponentResult;

FUNCTION SGSetSoundRecordChunkSize
                             (c: SGChannel; seconds: LongInt):
                             ComponentResult;

FUNCTION SGGetSoundRecordChunkSize
                             (c: SGChannel): LongInt;

FUNCTION SGSetSoundInputRate
                             (c: SGChannel; rate: Fixed): ComponentResult;

FUNCTION SGGetSoundInputRate
                             (c: SGChannel): Fixed;

FUNCTION SGSetSoundInputParameters
                             (c: SGChannel; sampleSize: Integer;
                             numChannels: Integer;
                             compressionType: OSType): ComponentResult;

FUNCTION SGGetSoundInputParameters
                             (c: SGChannel; VAR sampleSize: Integer;
                             VAR numChannels: Integer;
                             VAR compressionType: OSType): ComponentResult;

```

**Video Channel Callback Routines**

```

FUNCTION SGSetVideoBottlenecks
                             (c: SGChannel; VAR vb: VideoBottles):
                             ComponentResult;

```



## Sequence Grabber Components

```

FUNCTION SGGetVideoBottlenecks
    (c: SGChannel; VAR vb: VideoBottles):
        ComponentResult;

```

**Utility Routines for Video Channel Callback Functions**

```

FUNCTION SGGetBufferInfo    (c: SGChannel; bufferNum: Integer;
    VAR bufferPM: PixMapHandle;
    VAR bufferRect: Rect;
    VAR compressBuffer: GWorldPtr;
    VAR compressBufferRect: Rect): ComponentResult;

FUNCTION SGGrabFrame        (c: SGChannel; bufferNum: Integer):
    ComponentResult;

FUNCTION SGGrabFrameComplete
    (c: SGChannel; bufferNum: Integer;
    VAR done: Boolean): ComponentResult;

FUNCTION SGDisplayFrame     (c: SGChannel; bufferNum: Integer;
    mp: MatrixRecord; clipRgn: RgnHandle):
    ComponentResult;

FUNCTION SGCompressFrame    (c: SGChannel; bufferNum: Integer):
    ComponentResult;

FUNCTION SGCompressFrameComplete
    (c: SGChannel; bufferNum: Integer;
    VAR done: Boolean; VAR ci: SGCompressInfo):
    ComponentResult;

FUNCTION SGAddFrame         (c: SGChannel; bufferNum: Integer;
    atTime: TimeValue; scale: TimeScale;
    ci: SGCompressInfo): ComponentResult;

FUNCTION SGTransferFrameForCompress
    (c: SGChannel; bufferNum: Integer;
    mp: MatrixRecord; clipRgn: RgnHandle):
    ComponentResult;

FUNCTION SGGrabCompressComplete
    (c: SGChannel; VAR done: Boolean;
    VAR ci: SGCompressInfo; VAR tr: TimeRecord):
    ComponentResult;

FUNCTION SGDisplayCompress  (c: SGChannel; dataPtr: Ptr;
    desc: ImageDescriptionHandle;
    VAR mp: MatrixRecord;
    clipRgn: RgnHandle): ComponentResult;

```

Application-Defined Routines

---

```

FUNCTION MyGrabFunction      (c: SGChannel; bufferNum: Integer;
                             refCon: LongInt): ComponentResult;

FUNCTION MyGrabCompleteFunction
                             (c: SGChannel; bufferNum: Integer;
                              VAR done: Boolean; refCon: LongInt):
                              ComponentResult;

FUNCTION MyDisplayFunction   (c: SGChannel; bufferNum: Integer;
                              mp: MatrixRecord; clipRgn: RgnHandle;
                              refCon: LongInt): ComponentResult;

FUNCTION MyCompressFunction  (c: SGChannel; bufferNum: Integer;
                              refCon: LongInt): ComponentResult;

FUNCTION MyCompressCompleteFunction
                             (c: SGChannel; bufferNum: Integer;
                              VAR done: Boolean; VAR ci: SGCompressInfo;
                              refCon: LongInt): ComponentResult;

FUNCTION MyAddFrameFunction  (c: SGChannel; bufferNum: Integer;
                              atTime: TimeValue; scale: TimeScale;
                              ci: SGCompressInfo; refCon: LongInt):
                              ComponentResult;

FUNCTION MyTransferFrameFunction
                             (c: SGChannel; bufferNum: Integer;
                              mp: MatrixRecord; clipRgn: RgnHandle;
                              refCon: LongInt): ComponentResult;

FUNCTION MyGrabCompressCompleteFunction
                             (c: SGChannel; VAR done: Boolean;
                              VAR ci: SGCompressInfo; VAR tr: TimeRecord;
                              refCon: LongInt): ComponentResult;

FUNCTION MyDisplayCompressFunction
                             (c: SGChannel; dataPtr: Ptr;
                              desc: ImageDescriptionHandle;
                              VAR mp: MatrixRecord; clipRgn: RgnHandle;
                              refCon: LongInt): ComponentResult;

FUNCTION MyDataFunction      (c: SGChannel; p: Ptr; len: LongInt;
                              VAR offset: LongInt; chRefCon: LongInt;
                              time: TimeValue; writeType: Integer;
                              refCon: LongInt): OSErr;

FUNCTION MyModalFilter       (theDialog: DialogPtr;
                              VAR theEvent: EventRecord;
                              VAR ItemHit: Integer; refCon: LongInt): OSErr;

```

## Result Codes

---

|                             |       |  |
|-----------------------------|-------|--|
| noDeviceForChannel          | -9400 | Channel component cannot find its device                     |
| grabTimeComplete            | -9401 | Time limit for record operation has expired                  |
| cantDoThatInCurrentMode     | -9402 | Request invalid in current mode                              |
| notEnoughMemoryToGrab       | -9403 | Insufficient memory for record operation                     |
| notEnoughDiskSpaceToGrab    | -9404 | Insufficient disk space for record operation                 |
| couldntGetRequiredComponent | -9405 | Component not found  |
| badSGChannel                | -9406 | Invalid channel specified                                    |
| seqGrabInfoNotAvailable     | -9407 | Sequence grabber does not have this information at this time |
| deviceCantMeetRequest       | -9408 | Device cannot support grabber                                |

