

This chapter discusses sequence grabber channel components. **Sequence grabber channel components** manipulate captured data for sequence grabber components.

This chapter has been divided into the following sections:

- “About Sequence Grabber Channel Components” presents general information about sequence grabber channel components and their relationship to sequence grabber components.
- “Creating Sequence Grabber Channel Components” lists issues you should consider when developing a sequence grabber component, including required functions and the Component Manager result codes that you should use. It then provides a sample program that illustrates how to implement a sequence grabber channel component.
- “Using Sequence Grabber Channel Components” gives details on how sequence grabber components can use channel components to play captured data for the user or to save captured data in a QuickTime movie.
- “Sequence Grabber Channel Components Reference” describes the data structures and functions associated with the Apple-supplied sequence grabber channel component.
- “Summary of Sequence Grabber Channel Components” presents a summary of sequence grabber channel components in C and in Pascal.

If you are writing an application that uses the sequence grabber component, you do not need to read this chapter. Read the chapter “Sequence Grabber Components” in this book for a description of the services provided by sequence grabber components. If you are writing a sequence grabber channel component, you should read this chapter and read the earlier chapter that discusses sequence grabber components.

#### Note

Information in this chapter is presented from the perspective of a developer of a sequence grabber channel component. If you are developing a sequence grabber channel component, your component must support the interfaces described in this chapter. ♦

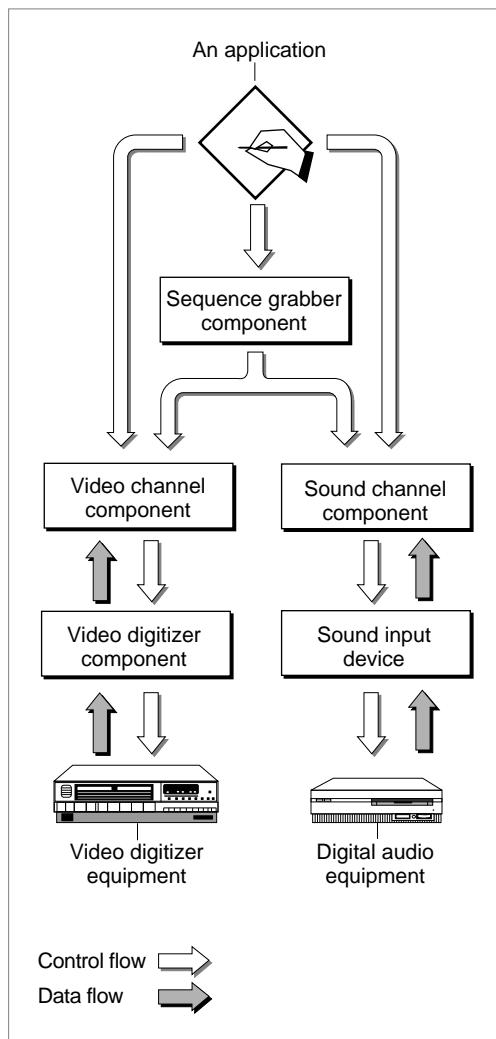
## About Sequence Grabber Channel Components

Sequence grabber components allow applications to obtain digitized data from sources that are external to a Macintosh computer. For example, applications can use a sequence grabber component to record video data from a video digitizer or a video disc player. The application can then request that the sequence grabber component store the captured video data in a QuickTime movie. In this manner users can acquire movie data from various sources. Applications can also use sequence grabber components to obtain and display data from external sources, without saving the captured data in a movie. For more information about sequence grabbers, see the chapter “Sequence Grabber Components” in this book.

## Sequence Grabber Channel Components

Sequence grabber components use sequence grabber channel components (or, simply, channel components) to obtain data from audio- or video-digitizing equipment. These components isolate the sequence grabber component from the details of working with the various types of data that can be collected. The functionality provided by a sequence grabber component depends upon 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 6-1 shows the relationship between these components and an application.

**Figure 6-1** Relationships of an 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 the chapter “Sequence Grabber Components” 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.

Note that sequence grabber channel components may support all of the functions that are supported by sequence grabber panel components. For example, sequence grabbers obtain settings information from a channel component by calling the channel component's `SGPanelGetSettings` function. See the chapter “Sequence Grabber Panel Components” in this book for more information about the sequence grabber configuration dialog box; the relationship between sequence grabbers, sequence grabber channels, and sequence grabber panels; and the functional interface supported by sequence grabber panel components.

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” 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 next section, “Creating Sequence Grabber Channel Components,” for more information about creating sequence grabber channel components.

## Creating Sequence Grabber Channel Components

---

Sequence grabber channel components are the most convenient mechanism for extending the ability of the sequence grabber component to accommodate new types of source data. For example, if you are developing special-purpose hardware that generates a new kind of data, you should create a channel component for that kind of data.

Refer to the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox* for a general discussion of how to create a component.

This section discusses issues you should consider when creating a sequence grabber channel component. It also provides a sample program for the implementation of a sequence grabber channel component.

## Component Type and Subtype Values

---

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

```
#define SeqGrabChannelType 'sgch';
```

Sequence grabber channel components use their component subtype value to indicate the media type created by the component. For example, a channel component that works with video data would have a subtype of 'vide' (this value is defined by the Movie Toolbox's `VideoMediaType` constant).

## Required Functions

---

At a minimum, your channel component should support the following functions:

<code>SGGetChannelInfo</code>	<code>SGRelease</code>
<code>SGGetChannelUsage</code>	<code>SGSetChannelRefCon</code>
<code>SGGetDataRate</code>	<code>SGSetChannelUsage</code>
<code>SGIdle</code>	<code>SGStartPreview</code>
<code>SGInitChannel</code>	<code>SGStartRecord</code>
<code>SGPause</code>	<code>SGStop</code>
<code>SGPrepare</code>	<code>SGWriteSamples</code>

In addition, if your channel component supports visual data, it should support at least the following functions:

```
SGGetChannelBounds
SGSetChannelBounds
SGSetGWorld
```

If your channel component supports audio data, it should support the following functions as well:

```
SGGetChannelVolume
SGSetChannelVolume
```

The remaining functions described in this section are optional. However, your channel component should support as many of these functions as possible, so that your component is more useful to applications and users.

## Component Manager Request Codes

As with all components, your channel component receives its requests from the Component Manager in the form of request codes. Apple strongly recommends that you fully support all of the Component Manager's request codes in your channel component—especially the target request. Developers will want to extend the capabilities of the sequence grabber channel components. The Component Manager's `CaptureComponent` function, which uses the target request, is the most convenient mechanism for obtaining the services of a component and then extending those services. If your channel component does not support the target request, then it cannot be used by applications or other components in this manner. You can use the following constants to refer to the request codes for each of the functions that your channel component must support.

```
/* basic sequence grabber channel component selectors */
kSGSetGWorldSelect          = 0x4;   /* SetGWorld */
kSGStartPreviewSelect       = 0x10;  /* SGStartPreview */
kSGStartRecordSelect        = 0x11;  /* SGStartRecord */
kSGIdleSelect               = 0x12;  /* SGIdle */
kSGStopSelect               = 0x13;  /* SGStop */
kSGPauseSelect              = 0x14;  /* SGPause */
kSGPrepareSelect             = 0x15;  /* SGPrepare */
kSGReleaseSelect            = 0x16;  /* SGRelease */
kSGUpdateSelect             = 0x27;  /* SGUpdate */

/* 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 */
```

## Sequence Grabber Channel Components

```

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 */

```

## Sequence Grabber Channel Components

```

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 channel control functions */
kSGCInitChannelSelect            = 0x180; /* SGCInitChannel */
kSGCWriteSamplesSelect           = 0x181; /* SGCWriteSamples */
kSGCGetDataRateSelect            = 0x182; /* SGCDataRate */
kSGCAlignChannelRectSelect       = 0x183; /* SGAlignChannelRect */
};

```

## A Sample Sequence Grabber Channel Component

---

This section describes a sample sequence grabber channel component for PICT image data.

### Implementing the Required Component Functions

---

Listing 6-1 supplies the component dispatchers for the sequence grabber channel component together with the required functions.

**Listing 6-1**      Setting up global variables and implementing required functions

```
#define kMediaTimeScale 600

typedef struct {
    ComponentInstance self;
    SeqGrabComponent grabber;
    long usage;
    Boolean paused;
    CGrafPtr destPort;
    GDHandle destGD;
    CGrafPort tempPort;
    MatrixRecord displayMatrix;
    Rect destRect;
    Rect srcRect;
    RgnHandle clip;
    Boolean inPreview;
    Boolean inRecord;
    TimeBase base;
    long bytesWritten;
    Boolean showTickCount;
    long saveUsage;
} SGPictGlobalsRecord, *SGPictGlobals;

pascal ComponentResult SGPICTDispatcher
    (ComponentParameters *params, Handle storage)
{
    OSErr err = badComponentSelector;
    ComponentFunction componentProc = 0;
```



## Sequence Grabber Channel Components

```

switch (params->what) {
    case kComponentOpenSelect:
        componentProc = SGPictOpen; break;
    case kComponentCloseSelect:
        componentProc = SGPictClose; break;
    case kComponentCanDoSelect:
        componentProc = SGPictCanDo; break;
    case kComponentVersionSelect:
        componentProc = SGPictVersion; break;
    case kSGSetGWorldSelect:
        componentProc = SGPictSetGWorld; break;
    case kSGStartPreviewSelect:
        componentProc = SGPictStartPreview; break;
    case kSGStartRecordSelect:
        componentProc = SGPictStartRecord; break;
    case kSGIdleSelect:
        componentProc = SGPictIdle; break;
    case kSGStopSelect:
        componentProc = SGPictStop; break;
    case kSGPauseSelect:
        componentProc = SGPictPause; break;
    case kSGPrepareSelect:
        componentProc = SGPictPrepare; break;
    case kSGReleaseSelect:
        componentProc = SGPictRelease; break;
    case kSGCSetChannelUsageSelect:
        componentProc = SGPictSetChannelUsage; break;
    case kSGCGetChannelUsageSelect:
        componentProc = SGPictGetChannelUsage; break;
    case kSGCSetChannelBoundsSelect:
        componentProc = SGPictSetChannelBounds; break;
    case kSGCGetChannelBoundsSelect:
        componentProc = SGPictGetChannelBounds; break;
    case kSGCGetChannelInfoSelect:
        componentProc = SGPictGetChannelInfo; break;
    case kSGCSetChannelMatrixSelect:
        componentProc = SGPictSetChannelMatrix; break;
    case kSGCGetChannelMatrixSelect:
        componentProc = SGPictGetChannelMatrix; break;
    case kSGCSetChannelClipSelect:
        componentProc = SGPictSetChannelClip; break;
    case kSGCGetChannelClipSelect:
        componentProc = SGPictGetChannelClip; break;
}

```

## Sequence Grabber Channel Components

```

        case kSGCGetChannelSampleDescriptionSelect:
            componentProc = SGPictGetChannelSampleDescription;
            break;
        case kSGCGetChannelDeviceListSelect:
            componentProc = SGPictGetChannelDeviceList; break;
        case kSGCSetChannelDeviceSelect:
            componentProc = SGPictSetChannelDevice; break;
        case kSGCGetChannelTimeScaleSelect:
            componentProc = SGPictGetChannelTimeScale; break;
        case kSGCInitChannelSelect:
            componentProc = SGPictInitChannel; break;
        case kSGCWriteSamplesSelect:
            componentProc = SGPictWriteSamples; break;
        case kSGCGetDataRateSelect:
            componentProc = SGPictGetDataRate; break;
        case kSGCPanelGetDitlSelect:
            componentProc = SGPictPanelGetDitl; break;
        case kSGCPanelInstallSelect:
            componentProc = SGPictPanelInstall; break;
        case kSGCPanelEventSelect:
            componentProc = SGPictPanelEvent; break;
        case kSGCPanelRemoveSelect:
            componentProc = SGPictPanelRemove; break;
        case kSGCPanelGetSettingsSelect:
            componentProc = SGPictPanelGetSettings; break;
        case kSGCPanelSetSettingsSelect:
            componentProc = SGPictPanelSetSettings; break;
        case 0x0100:
            componentProc = SGPictSetShowTickCount; break;
        case 0x0101:
            componentProc = SGPictGetShowTickCount; break;
    }

    if (componentProc)
        err = CallComponentFunctionWithStorage (storage, params,
                                                componentProc);

    return err;
}

pascal ComponentResult SGPictCanDo (SGPictGlobals store,
                                     short ftnNumber)
{
    switch (ftnNumber) {

```

## Sequence Grabber Channel Components

```

case kComponentOpenSelect:
case kComponentCloseSelect:
case kComponentCanDoSelect:
case kComponentVersionSelect:

case kSGSetGWorldSelect:

case kSGStartPreviewSelect:
case kSGStartRecordSelect:
case kSGIdleSelect:
case kSGStopSelect:
case kSGPauseSelect:
case kSGPrepareSelect:
case kSGReleaseSelect:

case kSGCSetChannelUsageSelect:
case kSGCGetChannelUsageSelect:
case kSGCSetChannelBoundsSelect:
case kSGCGetChannelBoundsSelect:
case kSGCGetChannelInfoSelect:

case kSGCSetChannelMatrixSelect:
case kSGCGetChannelMatrixSelect:
case kSGCSetChannelClipSelect:
case kSGCGetChannelClipSelect:

case kSGCGetChannelSampleDescriptionSelect:
case kSGCGetChannelDeviceListSelect:
case kSGCSetChannelDeviceSelect:
case kSGCGetChannelTimeScaleSelect:

case kSGCInitChannelSelect:
case kSGCWriteSamplesSelect:
case kSGCGetDataRateSelect:

case kSGCPanelGetDitlSelect:
case kSGCPanelInstallSelect:
case kSGCPanelEventSelect:
case kSGCPanelRemoveSelect:
case kSGCPanelGetSettingsSelect:
case kSGCPanelSetSettingsSelect:

/* private component functions */
case 0x0100:
case 0x0101:
    return true;

```

## Sequence Grabber Channel Components

```

        default:
            return false;
    }
}

pascal ComponentResult SGPictVersion (SGPictGlobals store)
{
    return 0x00020001;
}

pascal ComponentResult SGPictOpen (SGPictGlobals store,
                                    ComponentInstance self)
{
    OSErr err;
    GrafPtr savePort;

    /* allocate global variables */

    store =
    (SGPictGlobals)NewPtrClear(sizeof(SGPictGlobalsRecord));
    if (err = MemError()) goto bail;

    /* create a temporary port for drawing during the idle
       function */

    GetPort (&savePort);
    OpenCPort (&store->tempPort);
    SetPort ((GrafPtr)&store->tempPort);
    PortSize (4096, 4096);
    SetRectRgn (store->tempPort.visRgn, 0, 0, 4096, 4096);
    ClipRgn (store->tempPort.visRgn);
    SetPort (savePort);

    store->self = self;
    store->showTickCount = false;
    SetComponentInstanceStorage (self, (Handle)store);

bail:
    return err;
}

```

## Sequence Grabber Channel Components

```

pascal ComponentResult SGPictClose (SGPictGlobals store,
                                   ComponentInstance self)
{
    /* disposal operations */
    if (store) {
        if (store->clip) DisposeRgn(store->clip);
        CloseCPort(&store->tempPort);
        DisposPtr((Ptr)store);
    }

    return noErr;
}

```

### Initializing the Sequence Grabber Channel Component

---

To initialize the channel component, the sequence grabber component calls the `SGInitChannel` function, which is described on page 6-38.

The code in Listing 6-2 initializes channel variables. The grabber component calls the `SGPictInitChannel` function to initialize a sequence grabber channel component. The `SGPictInitChannel` function calls QuickDraw's `SetRect` routine and QuickTime's `SetIdentityMatrix` function to specify the size of the area (around a mouse-down event) in which the sequence grabber component will capture PICT images. For more on the `SetRect` routine, see the chapter “Basic QuickDraw” in *Inside Macintosh: Imaging*. For details on the `SetIdentityMatrix` function, see the chapter “Movie Toolbox” in *Inside Macintosh: QuickTime*.

---

**Listing 6-2**      Initializing the sequence grabber channel component

```

pascal ComponentResult SGPictInitChannel (SGPictGlobals store,
                                         SeqGrabComponent owner)
{
    /* initialize any variables here */
    SetRect(&store->srcRect, 0, 0, 160, 120); /* rectangle in which
                                                capture occurs */
    SetIdentityMatrix (&store->displayMatrix);

    store->grabber = owner;
    SGGetTimeBase (owner, &store->base);

    return noErr;
}

```

## Setting and Retrieving the Channel State

---

Listing 6-3 supplies configuration functions that set the usage parameters and storage for the channel component. (See the descriptions of the `SGSetChannelUsage` and `SGGetChannelUsage` functions on page 6-48 and page 6-49, respectively, for details.)

The sample code illustrates how to retrieve usage information. (See the description of the `SGGetChannelInfo` function on page 6-49 for details.) In this case, you indicate that the sequence grabber component has spatial boundaries by using the `seqGrabHasBounds` constant in the `channelInfo` parameter.

---

**Listing 6-3**      Determining usage parameters and getting usage data

```
pascal ComponentResult SGPictSetChannelUsage(SGPictGlobals store,
                                             long usage)
{
    /* remember usage */
    store->usage = usage;

    return noErr;
}

pascal ComponentResult SGPictGetChannelUsage(SGPictGlobals store,
                                             long *usage)
{
    /* return usage */
    *usage = store->usage;

    return noErr;
}

pascal ComponentResult SGPictGetChannelInfo (SGPictGlobals store,
                                             long *channelInfo)
{
    /* indicate that you have spatial boundaries */
    *channelInfo = seqGrabHasBounds;

    return noErr;
}
```

## Managing Spatial Properties

To set up an area in which the channel component displays image data, the sequence grabber should perform these tasks:

- Assign the destination graphics world and graphics device for the display of the captured image with the `SGSetGWorld` function (described on page 6-39).
- Specify a display transformation matrix for a video channel using the `SGSetChannelMatrix` function, which is described on page 6-57. Your function determines the matrix that is being set, validates it, and updates the matrix and destination rectangle. Your channel uses this matrix to transform its video image into the destination window.
- Obtain the channel's display transformation matrix by calling the `SGGetChannelMatrix` function, which is described on page 6-58.
- Specify the channel's display boundary rectangle with the `SGSetChannelBounds` function, which is described on page 6-63. The display boundary rectangle defines the destination for data from this channel and adjusts the channel matrix.
- Determine the channel's display boundary rectangle with the `SGGetChannelBounds` function (described on page 6-63).
- Dispose of the old clipping region and apply a new clipping region to the channel's display region using the `SGSetChannelClip` function, which is described on page 6-56.
- Retrieve the new clipping region by calling the `SGGetChannelClip` function (described on page 6-56).

The code in Listing 6-4 provides an example of how to manage the spatial characteristics of the area in which the channel component displays PICT image data.

**Listing 6-4** Managing spatial characteristics

```
pascal ComponentResult SGPictSetGWorld (SGPictGlobals store,
                                       CGrafPtr gp, GDHandle gd)
{
    /* remember the destination graphics world */
    store->destPort = gp;
    store->destGD = gd;

    return noErr;
}
```

## Sequence Grabber Channel Components

```

pascal ComponentResult SGPictSetChannelMatrix
    (SGPictGlobals store, const MatrixRecord *m)
{
    OSErr err = noErr;
    MatrixRecord mat;
    short matType;

    /* determine the matrix being set */
    if (m)
        mat = *m;
    else
        SetIdentityMatrix (&mat);

    /* validate it */
    matType = GetMatrixType (&mat);

    if ((mat.matrix[0][0] < 0) || (mat.matrix[1][1] < 0) ||
        (matType >= linearMatrixType))
        return paramErr;

    /* update the matrix and destination rectangle */
    store->displayMatrix = mat;
    store->destRect = store->srcRect;
    TransformRect (&mat, &store->destRect, nil);

    return err;
}

pascal ComponentResult SGPictGetChannelMatrix
    (SGPictGlobals store, MatrixRecord *m)
{
    /* return current matrix */
    *m = store->displayMatrix;

    return noErr;
}

pascal ComponentResult SGPictSetChannelBounds
    (SGPictGlobals store, const Rect *bounds)
{
    /* remember destination rect */
    store->destRect = *bounds;
}

```



## Sequence Grabber Channel Components

```

    /* recalculate display matrix from it */
    RectMatrix (&store->displayMatrix, &store->srcRect,
                &store->destRect);

    return noErr;
}

pascal ComponentResult SGPictGetChannelBounds
                                (SGPictGlobals store, Rect *bounds)
{
    /* return current boundaries */
    *bounds = store->destRect;

    return noErr;
}

pascal ComponentResult SGPictSetChannelClip (SGPictGlobals store,
                                              RgnHandle theClip)
{
    OSErr err = noErr;

    /* toss the old channel clipping */
    if (store->clip) {
        DisposeRgn (store->clip);
        store->clip = nil;
    }
    /* and remember the new one */
    if (theClip) {
        err = HandToHand ((Handle *)&theClip);
        store->clip = theClip;
    }

    return err;
}

pascal ComponentResult SGPictGetChannelClip
                                (SGPictGlobals store, RgnHandle *theClip)
{
    OSErr err = noErr;

    /* return clip, if there is one */
    if (*theClip = store->clip)
        err = HandToHand ((Handle *)theClip);
    return err;
}

```

## Controlling Previewing and Recording Operations

---

To preview and record image data in the channel component, the code in Listing 6-5 implements these tasks:

- The `SGStartPreview` function (described on page 6-40) instructs the channel to commence processing any source data. In preview mode, the component does not save any of the data it gathers from its source. Your channel component should immediately present the data to the user in the appropriate format for the channel's configuration and display video data in the destination display region.
- The `SGStartRecord` function (described on page 6-41) instructs the channel to begin recording data from its source. The sequence grabber component stores the collected data. The channel component should immediately begin recording data.
- The `SGIdle` function (described on page 6-42) allows the sequence grabber component to grant processing time to the channel component. The `SGIdle` function permits the processing time for the previewing and recording operations to take place. In the example shown in Listing 6-5, the work for the channel consists of getting the current time, adding data to the movie if recording, and showing the preview image if necessary.
- The `SGStop` function (described on page 6-43) stops the channel's preview and recording operations.
- The `SGPause` function (described on page 6-44) suspends or restarts the channel's preview and recording operations.
- The `SGPrepare` function (described on page 6-45) has the sequence grabber component prepare the channel for subsequent preview or record operations.
- The `SGRelease` function (described on page 6-46) releases any system resources that were allocated during preview or recording operations and that remain thereafter.

The code in Listing 6-5 illustrates a channel component's control of the previewing and recording of a PICT image.

---

**Listing 6-5** Controlling previewing and recording operations

```
pascal ComponentResult SGPictStartPreview (SGPictGlobals store)
{
    /* into preview mode */

    store->inPreview = (store->usage & seqGrabPreview) != 0;
    return noErr;
}

pascal ComponentResult SGPictStartRecord (SGPictGlobals store)
{
    /* into record mode (also preview, if PlayDuringRecord) */
    store->inRecord = (store->usage & seqGrabRecord) != 0;
    store->inPreview = (store->usage & seqGrabPlayDuringRecord) !=
```

## Sequence Grabber Channel Components

```

    0;
    return noErr;
}

pascal ComponentResult SGPictIdle (SGPictGlobals store)
{
    OSErr err = noErr;

    /* this is where the work for preview and record happens */
    if (!store->paused && (store->inRecord || store->inPreview)) {
        Point mouseLoc;
        Rect r;
        PicHandle tempPict = nil;
        TimeRecord tr;
        CGrafPtr savePort;
        GDHandle saveGD;
        Rect maxR;

        GetGWorld (&savePort, &saveGD);

        /* get the current time */
        GetTimeBaseTime (store->base, kMediaTimeScale, &tr);

        /* figure the current area around the mouse
           (only on main screen) */
        SetGWorld (&store->tempPort, GetMainDevice());
        GetMouse (&mouseLoc);
        LocalToGlobal (&mouseLoc);
        r.top = r.bottom = mouseLoc.v;
        r.left = r.right = mouseLoc.h;
        InsetRect(&r, -(store->srcRect.right >> 1),
                -(store->srcRect.bottom >> 1));
        maxR = (**GetMainDevice()).gdRect;
        if (r.left < maxR.left)
            OffsetRect (&r, -r.left + maxR.left, 0);
        if (r.top < maxR.top)
            OffsetRect (&r, 0, -r.top + maxR.top);
        if (r.right > maxR.right)
            OffsetRect(&r, maxR.right - r.right, 0);
        if (r.bottom > maxR.bottom)
            OffsetRect (&r, 0, maxR.bottom - r.bottom);

        /* copy the screen into a picture */
        tempPict = OpenPicture(&r);
        CopyBits ((BitMap *)&store->tempPort.portPixMap,

```

## Sequence Grabber Channel Components

```

        (BitMap *)&store->tempPort.portPixMap, &r, &r,
            srcCopy, nil);
    if (store->showTickCount) {
        /* if users want to see ticks, draw them */
        Str63 str;
        NumToString ( TickCount(), str);
        /* do some magic positioning */
        r.right = r.left + StringWidth(str) + 4;
        r.bottom = r.top + 14;
        EraseRect (&r);
        MoveTo(r.left + 2, r.bottom - 3);
        TextSize (12);
        DrawString (str);
    }
    ClosePicture();

    /* if recording, add data to movie */
    if (store->inRecord) {
        long offset;
        long pictSize = GetHandleSize ((Handle)tempPict);

        HLock ((Handle)tempPict);
        err = SGAddMovieData (store->grabber, store->self,
            (Ptr)*tempPict, pictSize, &offset, 0,
            tr.value.lo, seqGrabWriteAppend);
        store->bytesWritten += pictSize;
    }

    /* if you need to show the preview image, do that */
    if (store->inPreview) {
        RgnHandle saveClip;
        SetGWorld (store->destPort, store->destGD);
        if (store->clip) {
            saveClip = NewRgn();
            GetClip (saveClip);
            SetClip (store->clip);
        }
        DrawPicture (tempPict, &store->destRect);
        if (store->clip) {
            SetClip (saveClip);
            DisposeRgn (saveClip);
        }
    }
}

```

## Sequence Grabber Channel Components

```

        KillPicture (tempPict);

        SetGWorld (savePort, saveGD);
    }

    return err;
}

pascal ComponentResult SGPictStop (SGPictGlobals store)
{
    /* stop all previewing and recording */
    store->inRecord = store->inPreview = false;

    return noErr;
}

pascal ComponentResult SGPictPause (SGPictGlobals store,
                                     Byte pause)
{
    /* pause */
    store->paused = pause;

    return noErr;
}

pascal ComponentResult SGPictPrepare (SGPictGlobals store,
                                       Boolean prepareForPreview,
                                       Boolean prepareForRecord)
{
    /* prepare for previewing and recording operations--
       all you do here is initialize a variable */
    store->bytesWritten = 0;

    return noErr;
}

pascal ComponentResult SGPictRelease (SGPictGlobals store)
{
    /* no resources to release after previewing or recording */
    return noErr;
}

```

## Managing Channel Devices

---

To manage channel devices such as video digitizers or sound input drivers, you should

- let the sequence grabber retrieve a list of devices that are valid for the channel using the `SGGetChannelDeviceList` function (described on page 6-60)
- assign an appropriate channel device with the `SGSetChannelDevice` function (described on page 6-61)

Listing 6-6 provides examples of these required functions for channel device management. The `SGPictGetChannelDeviceList` function obtains a list of devices associated with the channel component. The `SGPictSetChannelDevice` function allows the sequence grabber to specify a channel device. In this code sample, there are no devices associated with the channel component.

---

**Listing 6-6** Coordinating devices for the channel component

```
pascal ComponentResult SGPictGetChannelDeviceList
    (SGPictGlobals store,
     long selectionFlags,
     SGDeviceList *list)
{
    *list = (SGDeviceList) NewHandleClear
        (sizeof (SGDeviceListRecord)); /* no devices */

    return MemError();
}

pascal ComponentResult SGPictSetChannelDevice
    (SGPictGlobals store, StringPtr name)
{
    /* you have no devices, so no problem */
    return noErr;
}
```

## Utility Functions for Recording Image Data

---

To record image data, the channel component must allow the sequence grabber to do the following:

- Obtain an appropriate time scale with the `SGGetChannelTimeScale` function (described on page 6-55).
- Retrieve the sample description of the image that is to be recorded with the `SGGetChannelSampleDescription` function (described on page 6-55).
- Create a track and media in which to record the sample image by calling the `SGWriteSamples` function (described on page 6-43). `SGWriteSamples` writes the captured data to a movie file after a record operation.

## Sequence Grabber Channel Components

- Obtain references from the sequence grabber and add them to the newly created media using the `SGGetNextFrameReference` function (described on page 6-88) so that the channel component can retrieve the sample references it stored.
- Determine how many bytes of captured data the channel is collecting each second using the `SGGetDataRate` function (described on page 6-54).

The code in Listing 6-7 shows how the channel component uses these utility functions to record PICT image data.

**Listing 6-7** Recording image data

```
pascal ComponentResult SGPictGetChannelTimeScale
                                (SGPictGlobals store, TimeScale *scale)
{
    *scale = kMediaTimeScale; /* a reasonable default time scale */

    return noErr;
}

pascal ComponentResult SGPictGetChannelSampleDescription
                                (SGPictGlobals store, Handle sampleDesc)
{
    OSErr err;
    SampleDescriptionPtr sdp;

    SetHandleSize (sampleDesc, sizeof(SampleDescription));
    if (err = MemError()) goto bail;

    /* make up a minimal sample description */
    sdp = (SampleDescriptionPtr)*sampleDesc;
    sdp->descSize = sizeof(SampleDescription);
    sdp->dataFormat = 'PICT';
    sdp->resvd1 = 0;
    sdp->resvd2 = 0;
    sdp->dataRefIndex = 0;

bail:
    return err;
}

pascal ComponentResult SGPictWriteSamples (SGPictGlobals store,
                                           Movie m, AliasHandle theFile)
{
    OSErr err = 0;
    Track pictT;
    Media pictM;
```

## Sequence Grabber Channel Components

```

long i;
MatrixRecord aMatrix;
Rect from, to;
seqGrabFrameInfo fi;
TimeRecord tr;
TimeValue mediaDuration;
SampleDescriptionHandle sampleDesc = 0;

/* after SGStop, this function creates the track and media */
if (!(store->usage & seqGrabRecord))
    return err;

/* get the sample description */
sampleDesc = (SampleDescriptionHandle)NewHandle(4);
if (err = MemError()) goto bail;
if (err = SGGetChannelSampleDescription (store->self,
                                         (Handle)sampleDesc)) goto bail;

/* figure out the track matrix */
SetRect (&from, 0, 0, store->srcRect.right,
        store->srcRect.bottom);
to = from;

TransformRect (&store->displayMatrix, &to, nil);

/* create the track and media */
pictT = NewMovieTrack (m, (long)from.right << 16,
                      (long)from.bottom << 16, 0);
pictM = NewTrackMedia (pictT, 'PICT', kMediaTimeScale,
                      (Handle)theFile, rAliasType);

/* spin in a loop getting sample references from the
   sequence grabber and adding them to the media */
fi.frameChannel = store->self;
i = -1;
do {
    TimeValue frameDuration;

    err = SGGetNextFrameReference (store->grabber,
                                   &fi, &frameDuration, &i);

    if (err) {
        if (err == paramErr)
            err = 0;
        break;
    }
}

```



## Sequence Grabber Channel Components

```

        err = AddMediaSampleReference (pictM,
                                       fi.frameOffset, fi.frameSize,
                                       frameDuration,
                                       sampleDesc, 1,
                                       0, 0);

        if (err == invalidDuration) {
            err = noErr;
            break;
        }
    } while (!err);

done:
    if (err) goto bail;

    GetTimeBaseTime (store->base, 0, &tr);
    ConvertTimeScale (&tr, kMediaTimeScale); /* trim media inserted
                                                to not extend
                                                beyond end time */

    mediaDuration = GetMediaDuration(pictM);

    /* add media to track */
    err = InsertMediaIntoTrack (pictT, 0, 0, tr.value.lo, kFix1);

    /* set track matrix */
    RectMatrix (&aMatrix, &from, &to);
    SetTrackMatrix (pictT, &aMatrix);

    /* set track clipping region */
    SetTrackClipRgn (pictT, store->clip);

bail:
    if (sampleDesc) DisposHandle ((Handle)sampleDesc);
    return err;
}

pascal ComponentResult SGPictGetDataRate (SGPictGlobals store,
                                           long *bytesPerSecond)
{
    /* take a guess at the data rate */
    *bytesPerSecond = 24 * 1024;
    if (store->bytesWritten) {
        TimeValue timeNow = GetTimeBaseTime (store->base, 8, nil);
        /* one-eighth second resolution */
    }
}

```

## Sequence Grabber Channel Components

```

        if (!timeNow)
            return seqGrabInfoNotAvailable;

        *bytesPerSecond = (store->bytesWritten / timeNow) * 8;
                        /* convert back to seconds */
    }
    return noErr;
}

```

## Providing Media-Specific Functions

---

The channel can provide media-specific functions for a particular channel type. These functions are analogous to the `SGSetVideoCompressorType` and `SGGetVideoCompressorType` functions (described on page 6-66 and page 6-67, respectively). These functions allow the sequence grabber to specify and determine the type of image compression the channel component is to apply to the captured video images.

The code in Listing 6-8 provides two specialized channel component functions, `SGPictSetShowTickCount` and `SGPictGetShowTickCount`, which set and retrieve the tick count, respectively. Note that both the functions refer to the `showTickCount` field in the `SGPictGlobals` structure.

---

**Listing 6-8**      Showing the tick count

```

pascal ComponentResult SGPictSetShowTickCount
                                (SGPictGlobals store, Boolean show)
{
    store->showTickCount = show;
    return noErr;
}

pascal ComponentResult SGPictGetShowTickCount
                                (SGPictGlobals store, Boolean *show)
{
    *show = store->showTickCount;
    return noErr;
}

```

## Managing the Settings Dialog Box

The channel allows the sequence grabber to manage the placement of your channel data in the sequence grabber's settings dialog box.

- To prepare to add the channel component's items to the settings dialog box, the sequence grabber obtains your item list by calling the sequence grabber panel component's `SGPanelGetDITL` function. It retrieves and detaches the dialog box template from the sequence grabber panel component.
- Once it has installed the items, the sequence grabber uses the `SGPanelInstall` function so initial values can be set. This function resets the channel to use the dialog window and preview mode. It also updates the boundaries to match the size of the user item list.
- To provide idle time in which to draw the channel's information in the settings dialog box, the sequence grabber uses the `SGPanelEvent` function. It allows the sequence grabber component to receive and process dialog events in a manner similar to a modal-dialog filter function. In this example, the information is the tick count.
- Prior to the removal of items from the settings dialog box, the sequence grabber component calls the `SGPanelRemove` function. The sequence grabber supplies information that specifies the channel that the panel is to configure, the dialog box, and the offset of the panel's items into the dialog box.

For details on the `SGPanelGetDITL`, `SGPanelInstall`, `SGPanelEvent`, and `SGPanelRemove` functions, see the chapter "Sequence Grabber Panel Components" in this book.

The code in Listing 6-9 calls the sequence grabber panel component and indicates that the channel component will display a tick count checkbox in the panel settings.

**Listing 6-9** Including a tick count checkbox in a dialog box in the panel component

```
pascal ComponentResult SGPictPanelGetDitl (SGPictGlobals store,
                                           Handle *ditl)
{
    /* get and detach your dialog template */
    *ditl = GetResource('DITL', 7000);
    if (!*ditl) return resNotFound;
    DetachResource(*ditl);
    return noErr;
}

pascal ComponentResult SGPictPanelInstall (SGPictGlobals store,
                                           SGChannel c,
                                           DialogPtr d,
                                           short itemOffset)
{
```

## Sequence Grabber Channel Components

```

    Rect newBounds;
    short kind;
    Handle h;

    /* reset this channel to use the dialog window and be in
       preview mode with no clip */
    SGSetGWorld (store->self, (CGrafPtr)d, GetMainDevice());
    SGGetChannelUsage (store->self, &store->saveUsage);
    SGSetChannelUsage (store->self, seqGrabPreview);
    SGSetChannelClip (c, nil);

    /* update boundaries to match size of user item */
    GetDItem (d, 1 + itemOffset, &kind, &h, &newBounds);
    SGSetChannelBounds (c, &newBounds);
    SGStartPreview (store->self);
    return noErr;
}

pascal ComponentResult SGPictPanelEvent (SGPictGlobals store,
                                         SGChannel c, DialogPtr d,
                                         short itemOffset,
                                         EventRecord *theEvent,
                                         short *itemHit,
                                         Boolean *handled)
{
    /* use idle time to draw */
    if (theEvent->what == nullEvent)
        return SGIdle (store->self);

    return noErr;
}

pascal ComponentResult SGPictPanelRemove (SGPictGlobals store,
                                           SGChannel c, DialogPtr d,
                                           short itemOffset)
{
    /* stop playing */
    SGStop (store->self);
    SGRelease (store->self);

    /* note that the clip and bounds are automatically restored
       for you because you stored them using the SGGetSettings
       function */

```

## Sequence Grabber Channel Components

```

    /* restore usage */
    SGSetChannelUsage(store->self, store->saveUsage);

    return noErr;
}

```

## Displaying Channel Information in the Settings Dialog Box

The final step in the implementation of a sequence grabber channel component is the display of the channel preview in the settings dialog box. Two sequence grabber functions, `SGSettingsDialog` and `SGGetSettingsDialog` (described in the chapter “Sequence Grabber Components” in this book), facilitate this process.

- The channel component instructs the sequence grabber to display its settings dialog box to the user by calling the sequence grabber component’s `SGSettingsDialog` function. The user can specify the configuration of a sequence grabber channel in this dialog box.
- To retrieve the current settings of all channels used by the sequence grabber, call the `SGGetSettings` function. The sequence grabber places all of this configuration information into a Movie Toolbox user data list.

Listing 6-10 provides code that creates a user data list to contain the tick count information for the sequence grabber’s settings dialog box, adds a matrix to the list, and stores clipping information (if any exists). The sample code then restores the clipping and the matrix.

**Listing 6-10**     Displaying channel settings

```

pascal ComponentResult SGPictPanelGetSettings
                                (SGPictGlobals store, SGChannel c,
                                UserData *result, long flags)
{
    OSErr err = noErr;
    UserData ud = 0;
    MatrixRecord matrix;
    RgnHandle clip;

    /* create a user data list to hold your state */

    if (err = NewUserData (&ud)) goto bail;

    /* add matrix to user data */

```

## Sequence Grabber Channel Components

```

    if (SGGetChannelMatrix (c, &matrix) == noErr) {
        if (err = SetUserDataItem (ud, &matrix, sizeof(matrix),
                                   sgMatrixType, 1))
            goto bail;
    }

    /* store clip, if there is one */
    if (SGGetChannelClip (c, &clip) == noErr) {
        if (clip)
            err = AddUserData (ud, (Handle)clip, sgClipType);

        else
            err = SetUserDataItem (ud, nil, 0, sgClipType, 1);
        /* add a dummy to indicate none */
        DisposeRgn(clip);
        if (err) goto bail;
    }

bail:
    if (err) {
        DisposeUserData (ud);
        ud = 0;
    }
    *result = ud;

    return err;
}

pascal ComponentResult SGPictPanelSetSettings
    (SGPictGlobals store,
     SGChannel c, UserData ud, long flags)
{
    OSErr err;
    RgnHandle clip = NewRgn();
    MatrixRecord matrix;

    /* restore clip, if one was stored */
    if (GetUserData (ud, (Handle)clip, sgClipType, 1) == noErr) {
        if (err = SGSetChannelClip
            (c, GetHandleSize ((Handle)clip) ? clip : 0))
            goto bail;
    }
}

```

## Sequence Grabber Channel Components

```

/* restore matrix */
if (err = GetUserDataItem (ud, &matrix, sizeof(matrix),
                           sgMatrixType, 1)) goto bail;
if (err = SGSetChannelMatrix (c, &matrix))
    goto bail;

bail:
    DisposeRgn (clip);
    return err;
}

```

## Using Sequence Grabber Channel Components

---

In response to application requests, sequence grabber components can use channel components in two ways: to play digitized data for the user or to save captured data in a QuickTime movie. The process of playing digitized data is called *previewing*; saving captured data in a movie is called *recording*. Applications can use previewing to allow the user to prepare to make a recording. Applications that use previewing can move directly from the preview operation to a record operation, without stopping the process.

The next two sections provide an overview of preview and record operations. A third section discusses the callback functions that are supported by some channel components.

### Previewing

---

Previewing captured data involves playing that data for the user as it is digitized. 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 the sequence grabber component follows to preview captured data.

1. First, the sequence grabber component opens a connection to your channel component, using the Component Manager's `OpenComponent` function. The sequence grabber component then calls your `SGInitChannel` function to initialize your component. For more on `SGInitChannel`, see page 6-38.
2. The sequence grabber component then configures your channel component for the preview operation. The `SGSetGWorld` function (described on page 6-39) sets the graphics world in which the preview is to be displayed. The `SGSetChannelUsage` function (described on page 6-48) specifies that your channel is to be used for previewing. The application can then use the appropriate channel configuration functions to prepare your channel for the preview operation. For video channels, it uses the functions discussed in "Configuration Functions for Video Channel Components" beginning on page 6-61. For sound channels, the sequence grabber uses the functions discussed in "Configuration Functions for Sound Channel Components" beginning on page 6-77.

## Sequence Grabber Channel Components

3. The sequence grabber component starts the preview operation by calling your `SGStartPreview` function (described on page 6-40). The sequence grabber component then begins collecting data from all of the channels participating in the preview and plays that data appropriately. The sequence grabber component can pause and restart the preview by calling the `SGPause` function (described on page 6-44). The sequence grabber component uses the `SGStop` function (described on page 6-43) to stop the preview. During the preview operation, the sequence grabber component calls your `SGIdle` function (described on page 6-42) frequently, so that your channel can perform its operation.
4. When the application is done previewing, the sequence grabber component can start recording or close its connection to your component.

## 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 component can also play the captured data for the user.

The following paragraphs discuss the steps the sequence grabber component follows to record captured data.

1. As with a preview operation, the sequence grabber component establishes a connection to your channel component by calling the Component Manager's `OpenComponent` function. It then initializes your component by calling your `SGInitChannel` function (described on page 6-38).
2. The sequence grabber component then configures your component for the record operation. The `SGSetGWorld` function (described on page 6-39) sets the graphics world in which the data is to be displayed. The `SGSetChannelUsage` function (described on page 6-48) specifies each channel that is to be used for recording. At this time, the sequence grabber component can also specify whether your component is to play its data while recording. The application can then use the appropriate channel configuration functions to prepare your channel for the record operation. For video channels, it uses the functions discussed in "Configuration Functions for Video Channel Components" beginning on page 6-61. For sound channels, the sequence grabber uses the functions discussed in "Configuration Functions for Sound Channel Components" beginning on page 6-77.
3. The sequence grabber component starts the record operation by calling your `SGStartRecord` function (described on page 6-41). The sequence grabber component then begins collecting data from the channels it has assigned, stores the data in a QuickTime movie, and, optionally, plays that data appropriately. The sequence grabber can pause and restart the record process by calling the `SGPause` function (described on page 6-44). During the record operation, the sequence grabber component calls your `SGIdle` function (described on page 6-42) frequently, so that your channel can perform its operation. The sequence grabber component uses the `SGStop` function (described on page 6-43) to stop the record operation. At this time,



your component saves the movie in the appropriate movie file if the sequence grabber component instructs your component to do so by calling your `SGWriteSamples` function (described on page 6-43).

4. When the application is done recording, it either returns to previewing or closes its connection to your component.

## Working With Callback Functions

---

Sequence grabber components provide callback functions that allow application developers to customize some aspects of capturing video data. It is your channel component's responsibility to call these callback functions at specified points in the data capture process. The application's function can then perform any special processing that is appropriate for the application. For example, an application can overlay text, such as a frame number, on each frame of video data as it is captured. These functions are discussed in detail in the next section.

### Note

Sound channel components do not support any callback functions. ♦

## Using Callback Functions for Video Channel Components

---

Sequence grabber components allow application developers to define a number of callback functions in their applications. Your channel component calls these functions at specific points in the process of collecting, compressing, and displaying the source visual data. By defining callback functions, a developer can control the process more precisely or customize the operation of the sequence grabber component and its channel components.

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

Note that your channel component need not call each and every callback function. If some functions are inappropriate to the operation of your channel, do not call them. However, if your component calls one function of a pair, be sure to call the other. For example, if your component calls an application's grab function, you must also call its grab-complete function.

The sequence grabber component uses the `SGSetVideoBottlenecks` function to assign callback functions to your video channel. The `SGGetVideoBottlenecks` function allows the sequence grabber to determine the callback functions that have been assigned to your video channel. See the chapter "Sequence Grabber Components" in this book for details on `SGSetVideoBottlenecks` and `SGGetVideoBottlenecks`.

## Sequence Grabber Channel Components

The following application-defined functions are supported by video channels and are described in the chapter “Sequence Grabber Components” in this book.

<code>MyAddFrameFunction</code>	<code>MyGrabCompressCompleteFunction</code>
<code>MyCompressCompleteFunction</code>	<code>MyGrabFunction</code>
<code>MyCompressFunction</code>	<code>MyTransferFrameFunction</code>
<code>MyDisplayFunction</code>	
<code>MyGrabCompleteFunction</code>	

## Using Utility Functions for Video Channel Component Callback Functions

---

Sequence grabber components provide a number of functions that application-defined functions can use. Several channel functions support those sequence grabber component functions.

The sequence grabber component uses the `SGGetBufferInfo` function to obtain information about a buffer that contains data to be manipulated by a callback function. Application callback functions can use the `SGGetBufferInfo` function to obtain information about a buffer that you have passed. This information is valid only during record operations, or after your channel has been prepared to record. The `SGGetBufferInfo` function is described in detail in the chapter “Sequence Grabber Components” in this book.

The following functions provide default behavior for application-defined grab, grab-complete, display, compress, compress-complete, add-frame, transfer-frame, display-compress, and grab-compress-complete functions:

- Your video channel component’s `SGGrabFrame` function provides the default behavior for an application’s grab function. Applications should call this function only from their grab function.
- Your channel component’s `SGGrabFrameComplete` function provides the default behavior for an application’s grab-complete function. Applications should call this function only from their grab-complete functions.
- Your channel component’s `SGDisplayFrame` function provides the default behavior for an application’s display function. Applications should call this function only from their display functions.
- Your video channel component’s `SGCompressFrame` function provides the default behavior for an application’s compress function. Applications should call this function only from their compress functions.
- Your channel component’s `SGCompressFrameComplete` function provides the default behavior for an application’s compress-complete function. Applications should call this function only from their compress-complete functions.
- Your component’s `SGAddFrame` function provides the default behavior for an application’s add-frame function. Applications should call this function only from their add-frame functions.

- Your component's `SGTransferFrameForCompress` function provides the default behavior for an application's transfer-frame function. Applications should call this function only from their transfer-frame functions.
- Your component's `SGGrabCompressComplete` function provides the default behavior for an application's grab-compress-complete function. Applications should call this function only from their grab-compress-complete function.
- Your component's `SGDisplayCompress` function provides the default behavior for an application's display-compress function. Applications should call this function only from their display-compress function.

## Sequence Grabber Channel Components Reference

---

This section describes the functions and associated data structures and constants that are specific to the Apple-supplied sequence grabber channel component. These functions are described from the perspective of a sequence grabber component—the most likely client of a sequence grabber channel component. If you are developing a sequence grabber channel component, your component must behave as described here.

### Functions

---

This section has been divided into the following topics:

- “Configuring Sequence Grabber Channel Components” describes the functions that allow sequence grabber components to configure your channel component.
- “Controlling Sequence Grabber Channel Components” discusses the functions that allow sequence grabber components to control your channel component.
- “Configuration Functions for All Channel Components” describes configuration functions that may be supported by all sequence grabber channel components.
- “Working With Channel Devices” discusses functions that allow the sequence grabber to assign devices to your channel.
- “Configuration Functions for Video Channel Components” describes functions that are supported only by video channel components.
- “Configuration Functions for Sound Channel Components” discusses functions that are supported only by sound channel components.
- “Utility Functions for Sequence Grabber Channel Components” describes several utility functions that sequence grabber components provide to sequence grabber channel components.

#### Note

If your channel component will also receive any of the functions defined in the interface for sequence grabber panel components, see the chapter “Sequence Grabber Panel Components” in this book for more information about these functions. ♦

## Configuring Sequence Grabber Channel Components

---

Sequence grabber components use a number of functions to establish the environment for grabbing or previewing digitized data. This section describes the channel component functions that allow the sequence grabber component to establish the environment for recording or previewing captured data.

The sequence grabber component uses the `SGInitChannel` function to initialize your channel prior to a record or preview operation.

The `SGSetGWorld` function allows the sequence grabber component to assign a graphics world to your component.

### SGInitChannel

---

A sequence grabber component calls the `SGInitChannel` function to initialize a sequence grabber channel component. Your component should perform its initialization processing here, rather than in response to the Component Manager's open request. The initialization processing may include allocating memory or checking for the availability of special-purpose hardware or software.

```
pascal ComponentResult SGInitChannel (SGChannel c,
                                     SeqGrabComponent owner);
```

<code>c</code>	Identifies the channel connection for this operation.
<code>owner</code>	Identifies the sequence grabber component that has connected to your channel component. You should save this value so that your channel component can call the utility functions that are provided by the sequence grabber component (see "Utility Functions for Sequence Grabber Channel Components," which begins on page 6-84, for information about these utility functions).

#### DESCRIPTION

If your component cannot gain access to the resources or equipment it needs to function properly, you should return a nonzero result code. If you return a nonzero result, the sequence grabber component closes its connection to your component and reports the error to the calling application.

#### RESULT CODES

<code>noDeviceForChannel</code>	-9400	Channel component cannot find its device
File Manager errors		
Memory Manager errors		

## SGSetGWorld

---

A sequence grabber component calls the `SGSetGWorld` function to establish the display environment for your channel component.

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

<code>s</code>	Identifies the sequence grabber component that has connected to your channel component.
<code>gp</code>	Specifies the destination graphics port. The sequence grabber component always sets this parameter to a valid value. The specified graphics port must be a color graphics port. The parameter is set to <code>nil</code> to use the current graphics port.
<code>gd</code>	Specifies the destination graphics device. The sequence grabber component always sets this parameter to a valid value.

### DESCRIPTION

Note that sequence grabber components may call this function for sound channel components as well as for video channel components.

### RESULT CODE

`cantDoThatInCurrentMode`    -9402    Request invalid in current mode

## Controlling Sequence Grabber Channel Components

---

Sequence grabber channel components must provide a full set of functions that allow the sequence grabber component to control the preview or record operation. The sequence grabber component can use these functions to start and stop the operation, to pause data collection, and to write captured data to a movie. This section describes these functions.

The sequence grabber component uses the `SGStartPreview` function to start a preview operation. The `SGStartRecord` function starts a record operation. The `SGStop` function stops your channel component after a preview or record operation.

The sequence grabber component grants processing time to your channel component by calling the `SGIdle` function. The sequence grabber notifies you of update events by calling your `SGUpdate` function.

The sequence grabber pauses the current operation by calling the `SGPause` function.

The sequence grabber component calls your `SGWriteSamples` function to write captured data to a movie file after a record operation.

## Sequence Grabber Channel Components

The sequence grabber component prepares your channel component for an upcoming preview or record operation by calling the `SGPrepare` function. This function also allows the sequence grabber component to verify that your component can support the parameters an application has specified. The `SGRelease` function releases system resources allocated during the `SGPrepare` function.

## SGStartPreview

---

The `SGStartPreview` function instructs your channel to begin processing its source data. In preview mode, your component does not save any of the data it gathers from its source.

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

**s**                      Identifies the sequence grabber component that has connected to your channel component.

### DESCRIPTION

Your channel component should immediately present the data to the user in the appropriate format, according to your channel's configuration (see "Configuration Functions for All Channel Components," which begins on page 6-46, for information about functions that configure channels). Display video data in the destination display region; play sound data 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

The sequence grabber component stops the preview process by calling your `SGStop` function, which is described on page 6-43.

## SGStartRecord

The `SGStartRecord` function instructs your channel component to begin recording data from its source. The sequence grabber component stores the collected data according to the recording parameters that the calling application specified with the sequence grabber component's `SGSetDataOutput` function (described in the chapter "Sequence Grabber Components" in this book). Your channel component should immediately begin recording data in the appropriate format, according to your channel's configuration (see "Configuration Functions for All Channel Components," which begins on page 6-46, for information about functions that configure channels).

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

**s**                      Identifies the sequence grabber component that has connected to your channel component.

### DESCRIPTION

The sequence grabber component can switch from previewing to recording by calling this function during a preview operation—the sequence grabber need not stop the preview operation first.

### RESULT CODES

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

File Manager errors  
Memory Manager errors

### SEE ALSO

The sequence grabber component stops the recording process by calling your `SGStop` function, which is described on page 6-43.

## SGIdle

---

The `SGIdle` function provides processing time to your channel component.

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

**s**                      Identifies the sequence grabber component that has connected to your channel component.

### DESCRIPTION

After starting a preview or record operation, the application calls the sequence grabber component's `SGIdle` function as often as possible. The sequence grabber component then calls your `SGIdle` function. This continues until the calling application stops the operation by calling the `SGStop` sequence grabber function.

Your `SGIdle` function reports several status and error conditions by means of its result code. If your component returns a nonzero result code during a record operation, the application should still call the `SGStop` function (described on page 6-43) so that the sequence grabber component can store the data it has collected.

### RESULT CODES

File Manager errors  
Memory Manager errors

## SGUpdate

---

The `SGUpdate` function allows you to learn about update events. This gives you an opportunity to update your display.

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

**s**                      Identifies the sequence grabber component that has connected to your channel component.

**updateRgn**           Indicates the part of the window that has been changed. This parameter specifies a portion of the window that has been changed. Applications can obtain this information by examining the appropriate window record. For example, they may call the sequence grabber in this manner:

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

If this parameter is set to `nil`, use the window's current visible region.



**DESCRIPTION**

Applications call the sequence grabber's `SGUpdate` function whenever they receive an update event for a window that contains a sequence grabber display. The sequence grabber then calls each affected channel. Applications should call this function before calling the Window Manager's `BeginUpdate` function.

**RESULT CODE**

`deviceCantMeetRequest`     **-9408**     Device cannot support grabber

**SGStop**

---

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

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

**s**                      Identifies the sequence grabber component that has connected to your channel component.

**DESCRIPTION**

In the case of a record operation, the sequence grabber component stores the collected movie data in the assigned movie file. The sequence grabber component then calls your `SGWriteSamples` function (described in the next section) to place the references to the captured data into the movie after it calls `SGStop`.

**▲ WARNING**

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

**RESULT CODES**

File Manager errors  
Memory Manager errors

**SGWriteSamples**

---

The sequence grabber component calls the `SGWriteSamples` function when it is ready to add recorded data to a movie.

```
pascal ComponentResult SGWriteSamples (SGChannel c, Movie m,
                                         AliasHandle theFile);
```

## Sequence Grabber Channel Components

<code>c</code>	Identifies the channel connection for this operation.
<code>m</code>	Identifies the movie to which your component should add the captured data. Your component should not make any other changes to the movie identified by this reference. Use the <code>SGWriteMovieData</code> function, described on page 6-86.
<code>theFile</code>	Identifies the movie file. The sequence grabber component provides this alias so that you can supply it to the Movie Toolbox. You should not open this file or write to it directly. Use the <code>SGWriteMovieData</code> function.

## DESCRIPTION

The sequence grabber component calls this function when the recording operation is complete, after calling your `SGStop` function (described on page 6-43). In this manner, your channel component can avoid unnecessary Movie Toolbox overhead during the record operation.

## SPECIAL CONSIDERATIONS

Your component should dispose of any buffered data and add the captured data to the movie. If necessary, use the Movie Toolbox's functions to create a track and a media. See the chapter "Movie Toolbox" in *Inside Macintosh: QuickTime* for details.

## RESULT CODES

File Manager errors  
Memory Manager errors

**SGPause**

A sequence grabber component can suspend or restart a record or preview operation by calling the `SGPause` function.

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

<code>s</code>	Identifies the sequence grabber component that has connected to your channel component.				
<code>pause</code>	Instructs your component to suspend or restart the current operation. The following values are valid: <table> <tr> <td><code>seqGrabUnpause</code></td><td>Restart the current operation.</td></tr> <tr> <td><code>seqGrabPause</code></td><td>Pause the current operation.</td></tr> </table>	<code>seqGrabUnpause</code>	Restart the current operation.	<code>seqGrabPause</code>	Pause the current operation.
<code>seqGrabUnpause</code>	Restart the current operation.				
<code>seqGrabPause</code>	Pause the current operation.				

DESCRIPTION

The sequence grabber component supplies a constant value in the `paused` parameter that instructs your component to pause or restart the current operation.

SPECIAL CONSIDERATIONS

Your component should not release any system resources or temporary memory associated with the current operation—you should be ready to restart the operation immediately.

RESULT CODES

`deviceCantMeetRequest`      -9408      Device cannot support grabber  
File Manager errors  
Memory Manager errors

SGPrepare

---

The `SGPrepare` function instructs your component to get ready to begin a preview or record operation (or both)—the sequence grabber component specifies the operations.

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

`s`                      Identifies the sequence grabber component that has connected to your channel component.

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

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

DESCRIPTION

Your component should do whatever is necessary to get ready to start the operation. The goal is to reduce the delay between the time when the sequence grabber calls your `SGStartPreview` function (described on page 6-40) or `SGStartRecord` function (described on page 6-41) and the time when the operation actually begins. This may involve allocating memory or readying special hardware.

**SPECIAL CONSIDERATIONS**

If the sequence grabber calls `SGPrepare` without subsequently starting a record or preview operation, it calls the `SGRelease` function (described in the next section) later. This allows your component to release any system resources it allocated during the `SGPrepare` function.

**RESULT CODES**

<code>paramErr</code>	-50	Invalid parameter specified
<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 your component to release any system resources it allocated during the `SGPrepare` function, which is described in the previous section.

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

**s** Identifies the sequence grabber component that has connected to your channel component.

**DESCRIPTION**

The sequence grabber component calls your `SGRelease` function whenever it calls `SGPrepare` without subsequently starting a record or preview operation.

**SPECIAL CONSIDERATIONS**

Note that the sequence grabber component may call `SGRelease` more than once after calling `SGPrepare`.

**Configuration Functions for All Channel Components**

---

Sequence grabber components use channel components to obtain digitized data from external media. Your channel is assigned to a sequence grabber component when the application calls the sequence grabber component's `SGNewChannel` function, described in the chapter "Sequence Grabber Components" in this book. The sequence grabber component must configure your channel before a preview or record operation. Your

channel component must provide a number of functions that allow the sequence grabber to configure the characteristics of your channel. Several of these functions work on any channel component. This section discusses these general channel configuration functions.

In addition, channel components provide functions that are specific to the channel type. The sequence grabber component supplied by Apple uses two types of channel components: video channel components and sound channel components. See “Configuration Functions for Video Channel Components,” which begins on page 6-61, for information about the configuration functions that work only with video channels. See “Configuration Functions for Sound Channel Components,” which begins on page 6-77, for information about the configuration functions that work only with sound channels.

The `SGSetChannelUsage` function specifies how your channel is to be used. The sequence grabber component can restrict a channel to use during record or preview operations. In addition, this function allows the sequence grabber component to specify whether your channel plays during a record operation. The `SGGetChannelUsage` function allows the sequence grabber component to determine a channel’s usage.

The `SGGetChannelInfo` function allows the sequence grabber component to determine some of the characteristics of your channel. For example, this function returns information indicating whether your channel has a visual or an audio representation.

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

The `SGSetChannelMaxFrames` function establishes a limit on the number of frames that your channel component will capture from a channel.

The `SGGetChannelMaxFrames` function enables the sequence grabber component to determine that limit.

The `SGSetChannelRefCon` function allows the sequence grabber component to set the value of a reference constant that your component passes to its callback functions (see “Using Callback Functions for Video Channel Components,” which begins on page 6-35, for information about the callback functions that are supported by video channels).

The `SGGetDataRate` function allows the sequence grabber component to determine how many bytes of captured data your channel is collecting each second.

The `SGGetChannelSampleDescription` function allows the sequence grabber to retrieve your channel’s sample description. The `SGGetChannelTimeScale` function allows it to obtain your channel’s time scale.

The sequence grabber can modify or retrieve your channel’s clipping region by calling the `SGSetChannelClip` or `SGGetChannelClip` function, respectively. The sequence grabber can work with your channel’s transformation matrix by calling the `SGSetChannelMatrix` and `SGGetChannelMatrix` functions.

## SGSetChannelUsage

---

The `SGSetChannelUsage` function specifies how your channel is to be used by the sequence grabber component.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>usage</code>	Contains flags specifying how your channel is to be used. The sequence grabber component may set more than one of these flags to 1. It sets unused flags to 0. The following flags are defined by the <code>SeqGrabUsageEnum</code> data type:
<code>seqGrabRecord</code>	Indicates that your channel is to be used during record operations. The sequence grabber component sets this flag to 1 to use a channel for recording.
<code>seqGrabPreview</code>	Indicates that your channel is to be used during preview operations. The sequence grabber component sets this flag to 1 to use a channel for previewing.
<code>seqGrabPlayDuringRecord</code>	Indicates that your component is to play its captured data during a record operation. If the sequence grabber component sets this flag to 1, your channel should play its captured data during a record operation, if the destination buffer is onscreen.

### DESCRIPTION

The sequence grabber component can specify that a channel is to be used for recording or previewing, or both. In addition, the sequence grabber component can control whether the data captured by a channel is displayed during the record or preview operation.

### RESULT CODES

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

## SGGetChannelUsage

---

The `SGGetChannelUsage` function allows the sequence grabber to determine how your channel is to be used.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>usage</code>	Contains a pointer to a location that is to receive flags specifying how your 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 by the <code>SeqGrabUsageEnum</code> data type:
<code>seqGrabRecord</code>	Indicates that your channel is to be used during record operations. Set this flag to 1 if your channel is being used for recording.
<code>seqGrabPreview</code>	Indicates that your channel is to be used during preview operations. Set this flag to 1 if your channel is being used for previewing.
<code>seqGrabPlayDuringRecord</code>	Indicates that your component is to play its captured data during a record operation. Set this flag to 1 if your channel plays its captured data during a record operation.

### SEE ALSO

The sequence grabber component establishes your channel's usage by calling your `SGSetChannelUsage` function, described in the previous section.

## SGGetChannelInfo

---

The `SGGetChannelInfo` function allows the sequence grabber 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);
```

<code>c</code>	Identifies the channel connection for this operation.
----------------	---

## Sequence Grabber Channel Components

`channelInfo`

Contains a pointer to a long integer that is to receive channel information flags. You may set more than one flag to 1. Set unused flags to 0. The following flags are defined:

`seqGrabHasBounds`

Indicates that your channel has a visual representation. If you set this flag to 1, the channel has a visual representation. The sequence grabber component may call your `SGSetChannelBounds` function (described on page 6-63).

`seqGrabHasVolume`

Indicates that your channel has an audio representation. If you set this flag to 1, the channel has an audio representation. The sequence grabber component may call your `SGSetChannelVolume` function (described on page 6-77).

`seqGrabHasDiscreteSamples`

Indicates that the data captured by your channel component is organized into discrete frames. If you set this flag to 1, the sequence grabber component may use the `SGSetChannelMaxFrames` function (described on page 6-52) to limit the number of frames processed in a record operation or the rate at which those frames are processed. If your channel's data is not organized into frames, set this flag to 0.

## SGSetChannelPlayFlags

---

The `SGSetChannelPlayFlags` function allows the sequence grabber component to influence the speed and quality with which your channel component displays data from its source.

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

`c` Identifies the channel connection for this operation.

`playFlags` Specifies a long integer that contains flags and values that influence channel playback. The following values are defined—the sequence grabber component must use one of these values:

`channelPlayNormal`

Instructs your channel component to use its default playback methodology.

`channelPlayFast`

Instructs your channel component to sacrifice playback quality in order to achieve the specified playback rate.



`channelPlayHighQuality`

Instructs your 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 to other programs in the computer. This option should not affect the quality of the recorded data, however.

The following flag is defined—the sequence grabber component may use this flag with any of the values defined for this parameter (unused flags are set to 0):

`channelPlayAllData`

Instructs your 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. The sequence grabber component sets this flag to 1 to play all the captured data. The sequence grabber component may combine this flag with any of the values defined for the `playFlags` parameter.

**DESCRIPTION**

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

**SGGetChannelPlayFlags**

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

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

`c` Identifies the channel connection for this operation.

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

`channelPlayNormal`

Your channel component uses its default playback methodology.

`channelPlayFast`

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

## Sequence Grabber Channel Components

`channelPlayHighQuality`

Your 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 to other programs in the computer. This option should not affect the quality of the recorded data, however.

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

`channelPlayAllData`

Your 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. The sequence grabber component sets this flag to 1 to play all the captured data. The sequence grabber component may combine this flag with any of the values defined for the `playFlags` parameter.

## SGSetChannelMaxFrames

---

The `SGSetChannelMaxFrames` function allows the sequence grabber to limit the number of frames that your channel component will capture during a record operation.

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

`c` Identifies the channel connection for this operation.

`frameCount` Specifies the maximum number of frames to capture during the preview or record operation. The sequence grabber component sets this parameter to -1 to remove the limit.

### DESCRIPTION

The `SGSetChannelMaxFrames` function works only with channels that have data that is organized into frames, such as video data from a video disc.

### RESULT CODES

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

**SEE ALSO**

You report whether your channel's data is organized into frames in your response to the `SGGetChannelInfo` function, which is described on page 6-49.

**SGGetChannelMaxFrames**

---

The `SGGetChannelMaxFrames` function allows the sequence grabber component to determine the number of frames left to be captured from your channel.

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

`c` Identifies the channel connection for this operation.

`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 channel component captures as many frames as it can.

**RESULT CODE**

`seqGrabInfoNotAvailable`    -9407    Channel component cannot support request

**SEE ALSO**

The sequence grabber component sets the starting value by calling the `SGSetChannelMaxFrames` function, which is described in the previous section.

**SGSetChannelRefCon**

---

The `SGSetChannelRefCon` function allows the sequence grabber component to set the value of a reference constant that your component passes to its callback functions.

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

`c` Identifies the channel connection for this operation.

`refCon` Specifies a reference constant value that your component should pass to the callback functions that have been assigned to this channel.

**DESCRIPTION**

Sound channels do not support callback functions.

**SEE ALSO**

See “Using Callback Functions for Video Channel Components,” which begins on page 6-36, for a description of the callback functions that are supported by video channel components.

**SGGetDataRate**

---

The sequence grabber component calls your component’s `SGGetDataRate` function in order to determine how much recording time is left. The sequence grabber calls your component when an application calls the sequence grabber component’s `SGGetTimeRemaining` function (see the chapter “Sequence Grabber Components” in this book for details).

```
pascal ComponentResult SGGetDataRate (SGChannel c,
                                     long *bytesPerSecond);
```

`c`                      Identifies the channel connection for this operation.

`bytesPerSecond`      Contains a pointer to a long integer that is to receive a value indicating the number of bytes your component is recording per second. Your component calculates this value based on its current operational parameters.

**DESCRIPTION**

Your component should calculate and return a value indicating the number of bytes of data your component is recording per second. The sequence grabber component uses this information, along with similar information gathered from other channels being used in the recording operation, to determine how many seconds of data may be recorded given the amount of space remaining.

**SPECIAL CONSIDERATIONS**

The sequence grabber component calls the `SGGetDataRate` function during the recording operation. Consequently, your component should service the request as quickly as possible.

## SGGetChannelSampleDescription

---

The `SGGetChannelSampleDescription` function allows the sequence grabber to retrieve your channel's sample description.

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

`c` Identifies the channel connection for this operation.

`sampleDesc` Specifies a handle that is to receive your sample description.

### DESCRIPTION

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

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

### RESULT CODE

`cantDoThatInCurrentMode`    -9402    Request invalid in current mode

## SGGetChannelTimeScale

---

The `SGGetChannelTimeScale` function allows the sequence grabber to retrieve your channel's time scale.

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

`c` Identifies the channel connection for this operation.

`scale` Contains a pointer to a time scale structure. Your channel component places information about its time scale into this structure.

### DESCRIPTION

The time scale you return typically corresponds to the time scale of the media that has been created by your channel. Applications may use this time scale in their data functions (see the chapter "Sequence Grabber Components" in this book for more information about application-defined data functions).

## SGSetChannelClip

---

The `SGSetChannelClip` function allows the sequence grabber to set your channel's clipping region.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>theClip</code>	Contains a handle to the new clipping region. You should make a copy of this region; the application may dispose of the region immediately. If this parameter is set to <code>nil</code> , remove the current clipping region.

### DESCRIPTION

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

## SGGetChannelClip

---

The `SGGetChannelClip` function allows the sequence grabber to retrieve your channel's clipping region.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>theClip</code>	Contains a pointer to a handle that is to receive the clipping region. The application is responsible for disposing of this handle. If there is no clipping region, set this handle to <code>nil</code> .

### Note

Some devices may not support clipping. ♦

## SGSetChannelMatrix

---

The `SGSetChannelMatrix` function allows the sequence grabber to set your channel's display transformation matrix.

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

**c** Identifies the channel connection for this operation.

**m** Contains a pointer to a matrix structure, as defined by the Movie Toolbox (see the chapter “Movie Toolbox” in *Inside Macintosh: QuickTime* for more information about matrix structures). This parameter is set to `nil` to select the identity matrix.

### DESCRIPTION

The `SGSetChannelMatrix` function allows the sequence grabber to specify a display transformation matrix for a video channel. Your channel uses this matrix to transform its video image into the destination window. If your channel cannot accommodate the matrix, return an appropriate result code. Note that the sequence grabber 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 (described on page 6-63). The `SGSetVideoRect` function modifies the matrix so that the specified video rectangle appears in the existing destination rectangle (see page 6-64 for more information about the `SGSetVideoRect` function).

### RESULT CODES

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

### SEE ALSO

The sequence grabber may retrieve your channel's matrix by calling the `SGGetChannelMatrix` function, which is discussed next.

## SGGetChannelMatrix

---

The `SGGetChannelMatrix` function allows the sequence grabber to retrieve your channel's display transformation matrix.

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

c	Identifies the channel connection for this operation.
m	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). Place your current matrix values into this matrix structure.

### SEE ALSO

The sequence grabber may set your 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 applications to determine the devices that can be, or the device that is, attached to a given sequence grabber channel. These devices, in turn, allow the channel component to control the digitizing equipment. For example, video channels use video digitizer components, and sound channels use sound input drivers. Applications can use these functions to present a list of available devices to the user, allowing the user to select a specific device for each channel. The sequence grabber passes these functions on to your channel component.

The sequence grabber may use the `SGGetChannelDeviceList` function to retrieve a list of devices that may be used by your channel.

The sequence grabber can use the `SGSetChannelDevice` function to assign a device to your channel.

The `SGGetChannelDeviceList` function uses 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;
```



## Sequence Grabber Channel Components

**Field descriptions**

count	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.
selectedIndex	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.
reserved	Reserved for Apple. Always set to 0.
entry	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.

```
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 applications may choose to present to the user. If the device does not support an icon, or if the sequence grabber chooses not to retrieve this information (by setting the <code>sgDeviceListWithIcons</code> flag to 0 when it calls the <code>SGGetChannelDeviceList</code> function, which is described in the next section), set this field to <code>nil</code> .
flags	Reflects the current status of the device. The following flag is defined: <div style="margin-left: 40px;"><code>sgDeviceNameFlagDeviceUnavailable</code>  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 the sequence grabber to retrieve a list of the devices that are valid for your channel.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>selectionFlags</code>	Controls the data you are to return for each device. The following flags are defined: <ul style="list-style-type: none"> <li><code>sgDeviceListWithIcons</code> <p>Specifies whether the sequence grabber wants to retrieve an icon for each device. If this flag is set to 1, return an icon for each device in the list, in the <code>icon</code> field. If this flag is set to 0, set the <code>icon</code> field to 0.</p> </li> <li><code>sgDeviceListDontCheckAvailability</code> <p>Controls whether you verify that each device is currently available. If this flag is set to 1, do not check the availability of each device. Otherwise, you should check each device's availability, and set the <code>sgDeviceNameFlagDeviceUnavailable</code> flag appropriately in each device name structure that you return.</p> </li> </ul>
<code>list</code>	Contains a pointer to a device list structure pointer. The channel creates a device name structure and returns a pointer to that structure in the field referred to by this parameter. Applications use the sequence grabber's <code>SGDisposeDeviceList</code> function to dispose of the memory used by the list.

### DESCRIPTION

This function allows the sequence grabber to retrieve a list of the devices that may be used by your channel. Each entry in this list identifies a valid device by name. Applications may then place these device names into a menu using the sequence grabber's `SGAppendDeviceListToMenu` function.

Applications may use this function in order to determine the device your channel is currently using. Be sure to set the `selectedIndex` field properly.

### RESULT CODES

Memory Manager errors

SEE ALSO

You may use the sequence grabber's `SGSortDeviceList` function to sort the entries in your device list structure. This function is discussed on page 6-89.

SGSetChannelDevice

The `SGSetChannelDevice` function allows the sequence grabber to assign a device to your channel.

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

- `c` Identifies the channel connection for this operation.
- `name` Contains a pointer to the device's name string. This name is contained in the `name` field of the appropriate device name structure in the device list that your channel returns to the `SGGetChannelDeviceList` function.

DESCRIPTION

When the sequence grabber calls your `SGSetChannelDevice` function, your channel should try to use the specified device instead of the device currently in use. The device name must be derived from your channel's device list.

RESULT CODES

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

SEE ALSO

The sequence grabber obtains the device list by calling your `SGGetChannelDeviceList` function, which is discussed in the previous section.

Configuration Functions for Video Channel Components

Video channel components provide a number of functions that allow the sequence grabber to configure the channel's video characteristics. This section describes these video channel configuration functions, which the sequence grabber component uses only with video channels.

The `SGSetChannelBounds` function allows the sequence grabber to set the display boundary rectangle for a video channel. The `SGGetChannelBounds` function determines a channel's boundary rectangle.

## Sequence Grabber Channel Components

The sequence grabber component uses the `SGGetSrcVideoBounds` function to determine the coordinates of the source video boundary rectangle. This rectangle defines the size of the source video image being captured by a video channel. The `SGSetVideoRect` function specifies a part of the source video boundary rectangle to be captured by the channel. The `SGGetVideoRect` function retrieves this active source video rectangle.

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

Video channel components 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 an application to work with a channel’s video digitizer component. Video channel components, in turn, must provide support for these functions. The sequence grabber component uses the `SGGetVideoDigitizerComponent` function to determine which video digitizer component is supplying data to your video channel component. The sequence grabber component sets a channel’s video digitizer component by calling the `SGSetVideoDigitizerComponent` function. If an application changes any video digitizer settings by calling the video digitizer component directly, the sequence grabber component informs your video channel component by calling the `SGVideoDigitizerChanged` function.

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. Some video channel components may provide functions that allow the sequence grabber component 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.

The sequence grabber can work with a video channel’s frame rate by calling the `SGSetFrameRate` and `SGGetFrameRate` functions. The sequence grabber can control whether your channel uses an offscreen buffer by calling your `SGSetUseScreenBuffer` and `SGGetUseScreenBuffer` functions.

Your `SGAlignChannelRect` function allows the sequence grabber to determine a channel’s optimum screen position.

## SGSetChannelBounds

---

The `SGSetChannelBounds` function allows the sequence grabber component to specify your channel's display boundary rectangle.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>bounds</code>	Contains a pointer to a rectangle that defines your channel's display boundary rectangle.

### DESCRIPTION

This rectangle defines the destination for data from this channel. This rectangle is defined in the graphics world that the sequence grabber component establishes by calling the `SGSetGWorld` function, described on page 6-39.

### SPECIAL CONSIDERATIONS

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

### 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

## SGGetChannelBounds

---

The `SGGetChannelBounds` function allows the sequence grabber component to determine your channel's display boundary rectangle.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>bounds</code>	Contains a pointer to a rectangle structure that is to receive information about your channel's display boundary rectangle.

**DESCRIPTION**

The sequence grabber component sets the boundary rectangle by calling the `SGSetChannelBounds` function, which is described in the previous section. This rectangle is defined in the graphics world that the sequence grabber establishes by calling the `SGSetGWorld` function, described on page 6-39.

**SGGetSrcVideoBounds**

---

The `SGGetSrcVideoBounds` function allows the sequence grabber component to determine the size of the source video boundary rectangle.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>r</code>	Contains a pointer to a rectangle structure that is to receive information about your channel's source video boundary rectangle.

**DESCRIPTION**

This rectangle defines the size of the source video image.

**RESULT CODE**

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

**SEE ALSO**

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 about video digitizer components).

**SGSetVideoRect**

---

The `SGSetVideoRect` function allows the sequence grabber component to specify a part of the source video image that is to be captured by your channel component.

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

<code>c</code>	Identifies the channel connection for this operation.
----------------	---

`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 the sequence grabber can obtain by calling the `SGGetSrcVideoBounds` function, described in the previous section.

DESCRIPTION

This rectangle must reside within the boundaries of the source video boundary rectangle. The sequence grabber component obtains the dimensions of the source video boundary rectangle by calling the `SGGetSrcVideoBounds` function. By default, your component should capture the entire video image, as defined by the source video boundary rectangle.

RESULT CODES

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

SEE ALSO

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.

SGGetVideoRect

---

The `SGGetVideoRect` function allows the sequence grabber to determine the portion of the source video image that your component is going to capture.

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

`c` Contains a pointer to the channel connection for this operation.

`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 your component is going to capture.

SEE ALSO

The sequence grabber uses the `SGSetVideoRect` function, which is described in the previous section, to set the dimensions of this rectangle.

## SGSetVideoCompressorType

---

The `SGSetVideoCompressorType` function allows the sequence grabber component to specify the type of image compression your component is to apply to the captured video images.

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

**c** Identifies the channel connection for this operation.

**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, its default compression type is selected.

### DESCRIPTION

In addition, your component should reset all image-compression parameters to their default values. The sequence grabber component can then use the `SGSetVideoCompressor` function, described on page 6-68, to change those compression parameters.

### 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 the sequence grabber component to determine the type of image compression that is being applied to your channel's video data.

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

**c** Identifies the channel connection for this operation.

**compressorType** Contains a pointer to an OSType field that is to receive information about the type of image compression to use. Return a value that corresponds to one of the image-compression 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, its default compression type is selected.

SEE ALSO

The sequence grabber component can set the image-compression type by calling the SGSetVideoCompressorType function, which is described in the previous section.

## SGSetVideoCompressor

---

The `SGSetVideoCompressor` function allows the sequence grabber component to specify many of the parameters that control image compression of the video data captured by your video channel.

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

<code>c</code>	Identifies the channel connection for this operation.
<code>depth</code>	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 the sequence grabber component sets this parameter to 0, let the sequence grabber component determine 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 component can determine which depths are supported by a given compressor by examining the compression 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).
<code>compressor</code>	Specifies the image compressor identifier. The sequence grabber component may specify a particular compressor by setting this parameter to its compressor identifier. You can obtain these identifiers from the Image Compression Manager's <code>GetCodecNameList</code> function.
<code>spatialQuality</code>	Specifies the desired quality of the compressed image. See the chapter "Image Compression Manager" in <i>Inside Macintosh: QuickTime</i> for valid values.
<code>temporalQuality</code>	Specifies the desired temporal quality of the sequence. This parameter governs the level of compression the sequence grabber component desires with respect to information in successive frames in the sequence. The sequence grabber component sets this parameter to 0 to prevent the image compressor from applying temporal compression to the sequence. See the chapter "Image Compression Manager" in <i>Inside Macintosh: QuickTime</i> 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. The sequence grabber component uses this parameter to control the frequency with 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).

RESULT CODES

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

SGGetVideoCompressor

The SGGetVideoCompressor function allows the sequence grabber component to determine your channel’s current image-compression parameters.

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

c	Identifies the channel connection for this operation.
depth	Contains a pointer to a field that is to receive the depth at which the image is likely to be viewed. Image compressor components may use the depth as an indication of the color or grayscale resolution of the compressed images. Return the depth value currently in use by your channel component. If this parameter is set to nil, the sequence grabber component is not interested in this information.

## Sequence Grabber Channel Components

`compressor`

Contains a pointer to a field that is to receive an image compressor identifier. Return the identifier that corresponds to the image compressor your channel is using. If this parameter is set to `nil`, the sequence grabber component is not interested in this information.

`spatialQuality`

Contains a pointer to a field that is to receive the desired compressed image quality. Return the current quality value. See the chapter “Image Compression Manager” in *Inside Macintosh: QuickTime* for valid values. If this parameter is set to `nil`, the sequence grabber component is not interested in this information.

`temporalQuality`

Contains a pointer to a field that is to receive the desired temporal quality of the sequence. This parameter governs the level of compression you desire with respect to information between successive frames in the sequence. Return the current temporal quality value. If this parameter is set to `nil`, the sequence grabber component is not interested in this information.

`keyFrameRate`

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. Return the current key frame rate. If this parameter is set to `nil`, the sequence grabber component is not interested in this information.

## SEE ALSO

The sequence grabber component can set these parameters by calling the `SGSetVideoCompressor` function, which is described in the previous section.

**SGSetVideoDigitizerComponent**

---

The `SGSetVideoDigitizerComponent` function allows the sequence grabber component to assign a video digitizer component to your video channel.

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

`c`                      Identifies the channel connection for this operation.

## Sequence Grabber Channel Components

**vdig** Contains a component instance that identifies a connection to a video digitizer component. Your video channel component should use this video digitizer component to obtain its source video data.

**DESCRIPTION**

Typically, your video channel component locates its own video digitizer component. The sequence grabber component calls the `SGSetVideoDigitizerComponent` function if an application chooses to assign a video digitizer to a video channel.

**RESULT CODE**

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

**SGGetVideoDigitizerComponent**

---

The `SGGetVideoDigitizerComponent` function allows the sequence grabber component to determine the video digitizer component that is providing source video to your video channel component. For example, the sequence grabber component can use this function to obtain access to the video digitizer component so that the grabber component can set the digitizer's parameters. See the chapter "Video Digitizer Components" in this book for information about video digitizer components.

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

**c** Identifies the channel connection for this operation.

**DESCRIPTION**

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

**SEE ALSO**

If the sequence grabber component changes any video digitizer component parameters, it notifies your sequence grabber channel component by calling your `SGVideoDigitizerChanged` function, which is described in the next section.

## SGVideoDigitizerChanged

---

The `SGVideoDigitizerChanged` function allows the sequence grabber component to notify your component whenever an application changes the configuration of your video channel's video digitizer.

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

`c`                      Identifies the channel connection for this operation.

### DESCRIPTION

You should update any status information you maintain regarding the video digitizer component used by your channel component.

### 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 allows the sequence grabber component to direct your component to create a filter buffer for your video channel. Logically, this buffer sits between the source video buffer and the destination rectangle that the sequence grabber component sets with the `SGSetChannelBounds` function, described on page 6-63. The filter buffer should be larger than the area enclosed by the destination rectangle.

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

`c`                      Identifies the channel connection for this operation.

`depth`                Specifies the pixel depth of the filter buffer. If the sequence grabber sets this parameter to 0, use the depth of the video buffer (which the sequence grabber sets with the `SGSetChannelBounds` function).

`compressSize`

Contains a pointer to the dimensions of the filter buffer. This buffer should be larger than the destination buffer. The sequence grabber component sets this parameter to `nil`, or it sets the coordinates of this rectangle to 0 (specifying an empty rectangle), to stop filtering the input source video data.

#### DESCRIPTION

If the sequence grabber component establishes a filter buffer for a channel, your channel component should place its captured video image into the filter buffer and then copy the image into the destination buffer. This process may be too slow for some record operations, but it can be useful during controlled record operations (where the source video can be read on a frame-by-frame basis).

#### RESULT CODE

`cantDoThatInCurrentMode`      -9402      Request invalid in current mode

### SGGetCompressBuffer

---

The `SGGetCompressBuffer` function returns information about the filter buffer that the sequence grabber component has established for your video channel.

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

`c`      Identifies the channel connection for this operation.

`depth`      Contains a pointer to a field that is to receive the pixel depth of the filter buffer. If your component is not filtering the input video data, set the returned value to 0.

`compressSize`

Contains a pointer to a rectangle structure that is to receive the dimensions of the filter buffer. If your component is not filtering the input video data, return an empty rectangle (all coordinates set to 0).

#### SEE ALSO

The sequence grabber component sets a filter buffer by calling the `SGSetCompressBuffer` function, which is described in the previous section.

## SGSetFrameRate

---

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

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

`c` Identifies the channel connection for this operation.

`frameRate` Specifies the desired frame rate. If this parameter is set to 0, use your channel's default frame rate. Typically, this corresponds to the fastest rate that your channel can support.

### DESCRIPTION

The `SGSetFrameRate` function allows the sequence grabber to control a video channel's frame rate. Note that the digitizing hardware may not be able to support the full rate that the sequence grabber specifies. If the rate is too high, operate at the highest rate you can support.

### SPECIAL CONSIDERATIONS

Note that this function will not be called when you are recording.

### RESULT CODES

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

### SEE ALSO

The sequence grabber can retrieve your channel's current frame rate by calling your `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);
```

`c` Identifies the channel connection for this operation.

`frameRate` Contains a pointer to a field to receive the current frame rate. Return your channel's current frame rate.



DESCRIPTION

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

SEE ALSO

The sequence grabber can set your channel's frame rate by calling the `SGSetFrameRate` function, which is described in the previous section.

SGSetUseScreenBuffer

The `SGSetUseScreenBuffer` function allows the sequence grabber to control whether your video channel uses an offscreen buffer.

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

- `c` Identifies the channel connection for this operation.
- `useScreenBuffer` Indicates whether to use an offscreen buffer. If this parameter is set to `true`, draw directly to the screen. If it is set to `false`, your channel may use an offscreen buffer. If your channel cannot work with offscreen buffers, ignore this parameter.

DESCRIPTION

By default, video channels try to draw directly to the screen. The `SGSetUseScreenBuffer` function allows the sequence grabber to direct your video channel to draw to an offscreen buffer. If your channel cannot draw offscreen, ignore this function. Note that this function will not be called when you are recording.

RESULT CODES

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

SEE ALSO

The sequence grabber can determine whether it has allowed your channel to draw offscreen by calling your `SGGetUseScreenBuffer` function, which is described next.

## SGGetUseScreenBuffer

---

The `SGGetUseScreenBuffer` function allows the sequence grabber to determine whether your video channel is allowed to use an offscreen buffer.

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

**c** Identifies the channel connection for this operation.

**useScreenBuffer**

Contains a pointer to a Boolean value. Set this field to `true` if your channel draws directly to the screen. Set it to `false` if your channel can use an offscreen buffer. If your channel cannot work with offscreen buffers, ignore this value.

### DESCRIPTION

By default, video channels draw directly to the screen. The sequence grabber can direct a channel to draw to an offscreen buffer by calling your `SGSetUseScreenBuffer` function. If the channels can work offscreen, it then allocates and draws 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.

## SAlignChannelRect

---

The sequence grabber calls your `SAlignChannelRect` function in order to determine whether your channel prefers to draw at a particular screen location.

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

**c** Identifies the connection to your channel.

**r**

Contains a pointer to a rectangle. On entry, this rectangle contains coordinates at which the sequence grabber would like to draw your captured video image. If your component can draw more efficiently or at a higher frame rate at a different location, update the contents of this rectangle to reflect where you would prefer to draw. The rectangle will be passed in with global, not local, coordinates.

DESCRIPTION

The sequence grabber uses your `SGAlignChannelRect` function to determine the best alignment for your captured image.

RESULT CODE

`badComponentSelector`      `0x80008002`      Function not supported

Configuration Functions for Sound Channel Components

---

Sound channel components provide a number of functions that allow sequence grabber components to configure the component's sound channel. This section describes these sound channel configuration functions. The sequence grabber component uses these functions only with sound channels.

The `SGSetChannelVolume` function allows the sequence grabber component to control a channel's sound volume. The sequence grabber component uses the `SGGetChannelVolume` function to determine a channel's volume.

The `SGSetSoundInputDriver` specifies a channel's sound input device. The sequence grabber component can determine a channel's sound input device by calling the `SGGetSoundInputDriver` function. If an application changes any attributes of the sound input device, the sequence grabber component notifies your sound component by calling the `SGSoundInputDriverChanged` function.

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

The sequence grabber component controls the rate at which your sound channel samples the input data by calling the `SGSetSoundInputRate` function. The sequence grabber component can determine the sample rate by calling the `SGGetSoundInputRate` function.

The sequence grabber can control other sound input parameters by using your `SGSetSoundInputParameters` and `SGGetSoundInputParameters` functions.

SGSetChannelVolume

---

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

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

`c`                      Identifies the channel connection for this operation.

## Sequence Grabber Channel Components

**volume** Specifies the volume setting of your 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**

Use this volume setting during playback—this setting should not affect the record level or the volume of the track in the recorded QuickTime movie.

**SGGetChannelVolume**

---

The `SGGetChannelVolume` function allows the sequence grabber component to determine your channel's sound volume setting.

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

**c** Identifies the channel connection for this operation.

**volume** 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**

The sequence grabber component establishes the volume setting by calling the `SGSetChannelVolume` function, described in the previous section.

**SGSetSoundInputDriver**

---

Some sound channel components may use sound input devices to obtain their source data. The `SGSetSoundInputDriver` function allows the sequence grabber component to assign a sound input device to your sound channel.

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

**c** Identifies the channel connection for this operation.

## Sequence Grabber Channel Components

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 your sound channel component does not use sound input devices, return a nonzero result code.

**RESULT CODES**

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

**SEE ALSO**

For more information about sound input devices, see *Inside Macintosh: More Macintosh Toolbox*—in particular, refer to the discussion of the `SPBGetIndexedDevice` function in the chapter “Sound Manager.”

**SGGetSoundInputDriver**

---

The `SGGetSoundInputDriver` function allows the sequence grabber component to determine the sound input device currently in use by your sound channel component.

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

`c`                      Identifies the channel connection for this operation.

**DESCRIPTION**

The sequence grabber component may want to gain access to the sound input device if it wants to change the device’s configuration. For example, the sequence grabber component may want to configure the device for stereo sound. If the sequence grabber component changes any of the device’s operating parameters, it informs your sequence grabber channel component by calling your `SGSoundInputDriverChanged` function, which is described in the next section.

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

**SEE ALSO**

The sequence grabber component can assign a sound input device to a sound channel by calling the `SGSetSoundInputDriver` function, which is described in the previous section.

## SGSoundInputDriverChanged

---

The `SGSoundInputDriverChanged` function allows the sequence grabber component to notify your sound channel component whenever an application changes the configuration of your sound channel's sound input device.

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

`c`                      Identifies the channel connection for this operation.

**DESCRIPTION**

Your component should update any sound device status information it maintains.

## SGSetSoundRecordChunkSize

---

During record operations, the sequence grabber component and its sound channels work 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.

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

`c`                      Identifies the channel connection for this operation.

`seconds`              Specifies the number of seconds of sound data your sound channel component is to work with at a time. This parameter is set to a negative fixed-point number to specify a fraction of a second. For example, to set the duration to half a second, `-0.5` is passed in.

**DESCRIPTION**

The sequence grabber component can control the amount of sound data in each chunk by calling the `SGSetSoundRecordChunkSize` function. The sequence grabber component specifies the number of seconds of sound data your channel is to work with at a time.

SPECIAL CONSIDERATIONS

The `SGSetSoundRecordChunkSize` function may return a fraction of a second (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 the sequence grabber component to determine the amount of sound data your sound channel component works with at a time.

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

`c` Identifies the channel connection for this operation.

DESCRIPTION

The `SGGetSoundRecordChunkSize` function returns a long integer that specifies the number of seconds of sound data your channel works with at a time.

SEE ALSO

The sequence grabber component sets this value by calling the `SGSetSoundRecordChunkSize` function, which is described in the previous section.

SGSetSoundInputRate

---

The `SGSetSoundInputRate` function allows the sequence grabber component to set the rate at which your sound channel obtains its sound data.

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

`c` Identifies the channel connection for this operation.

`rate` Specifies the rate at which your sound channel is to acquire data. This parameter specifies the number of samples your sound channel is to generate per second. If your sound channel cannot support the specified rate, use the closest available rate that you can support. If this parameter is set to 0, use your default rate.

**RESULT CODES**

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

**SGGetSoundInputRate**

---

The `SGGetSoundInputRate` function allows the sequence grabber component to determine the rate at which your sound channel is collecting sound data.

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

`c` Identifies the channel connection for this operation.

**DESCRIPTION**

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

**SEE ALSO**

The sequence grabber component sets this rate by calling the `SGSetSoundInputRate` function, which is described in the previous section.

**SGSetSoundInputParameters**

---

The `SGSetSoundInputParameters` function allows the sequence grabber to set some parameters that relate to sound recording.

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

`c` Identifies the channel connection for this operation.

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

`numChannels` Indicates the number of sound channels to be used by the sound sample. This field is set to 1 for monaural sounds; it is set to 2 for stereo sounds.



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

Sequence grabbers may use the `SGSetSoundInputParameters` function to control many parameters relating to sound recording. All of the sound parameters support two special values. If any of these parameters are set to 0, your channel should not change the current value of that parameter. If any are set to -1, return that parameter to its default value.

If your sound device cannot support a specified parameter value, return an appropriate Sound Manager result code.

**RESULT CODES**

Sound Manager errors

**SGGetSoundInputParameters**

The `SGGetSoundInputParameters` function allows the sequence grabber to retrieve some parameters that relate to sound recording.

<code>pascal ComponentResult SGGetSoundInputParameters (SGChannel c,</code>	
<code>short *sampleSize,</code>	
<code>short *numChannels,</code>	
<code>OSType *compressionType);</code>	
<code>c</code>	Identifies the channel connection for this operation.
<code>sampleSize</code>	Contains a pointer to a field to receive the sample size. Set this field to 8 for 8-bit sound; set the field to 16 for 16-bit sound.
<code>numChannels</code>	Contains a pointer to a field to receive the number of sound channels used by the sound sample. Set this field to 1 for monaural sounds; set the field to 2 for stereo sounds.

## Sequence Grabber Channel Components

`compressionType`

Contains a pointer to a field to receive the format of the sound data. You may return the following values:

<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**

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

## Utility Functions for Sequence Grabber Channel Components

---

Sequence grabber components provide several utility functions that your channel component can use. This section discusses those functions.

The `SGAddMovieData` function lets you add data and sample references to a movie.

Alternatively, you can use the `SGWriteMovieData` function to add data to a movie, and the `SGAddFrameReference` and `SGGetNextFrameReference` functions to keep track of sample references prior to creating a QuickTime movie from recorded data.

The `SGSortDeviceList` function allows you to sort the entries in the device list that you create for the sequence grabber when it calls your `SGGetChannelDeviceList` function (which is discussed on page 6-60).

The `SGChangedSource` function allows you to tell the sequence grabber that you have changed your source device.

The `SGAddFrameReference` and `SGGetNextFrameReference` functions take a pointer to a frame information structure as a parameter. 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. Your channel component obtains this value from the SGWriteMovieData function, described on page 6-86.
frameTime	Specifies the time at which your 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, your 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 your channel.
frameRefCon	Contains a reference constant for use by your channel component. You can use this value in any way that is appropriate for your channel component. For example, video channel components may use this value to store a reference to frame differencing information for a temporally compressed image sequence.

SGAddMovieData

The SGAddMovieData function allows your channel component to add data to a movie. This function combines the services provided by the SGWriteMovieData and SGAddFrameReference functions. Your channel component should not write data directly to the movie file—use this function instead.

```
pascal ComponentResult SGAddMovieData (SeqGrabComponent s,
                                         SGChannel c, Ptr p,
                                         long len,
                                         long *offset,
                                         long chRefCon,
                                         TimeValue time,
                                         short writeType);
```

s	Specifies the component instance that identifies the sequence grabber component that is using your channel. The sequence grabber provides this to you when it calls your SGInitChannel function (described on page 6-38).
c	Identifies the connection to your channel.
p	Specifies the location of the data to be added to the movie.
len	Indicates the number of bytes of data to be added to the movie.

## Sequence Grabber Channel Components

<code>offset</code>	Contains a pointer to a field that is to receive the offset to the new data in the movie. The sequence grabber component returns an offset that is correct in the context of the movie resource, even if the movie is currently stored in memory. That is, if the movie is in memory, the returned offset reflects the location that the data will have in a movie on a permanent storage device, such as a disk.
<code>chRefCon</code>	Contains your channel's reference constant.
<code>time</code>	Specifies the time at which your channel captured the frame. This time value is expressed in your channel's time scale.
<code>writeType</code>	Specifies the type of write operation. The following values are valid: <ul style="list-style-type: none"> <li><code>seqGrabWriteAppend</code> Append the new data to the end of the file. The sequence grabber sets the field referred to by the <code>offset</code> parameter to reflect the location at which it added the data.</li> <li><code>seqGrabWriteReserve</code> 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. The sequence grabber sets the field referred to by the <code>offset</code> parameter to the location of the reserved space.</li> <li><code>seqGrabWriteFill</code> Write the data into the location specified by the field referred to by the <code>offset</code> parameter. The sequence grabber sets that field to the location of the byte following the last byte it wrote.  This option is used to fill the space reserved previously when the <code>writeType</code> parameter was set to <code>seqGrabWriteReserve</code>.</li> </ul>

## RESULT CODES

File Manager errors  
Memory Manager errors

**SGWriteMovieData**

---

The `SGWriteMovieData` function allows your channel component to add data to a movie.

```
pascal ComponentResult SGWriteMovieData (SeqGrabComponent s,
                                          SGChannel c, Ptr p,
                                          long len, long *offset);
```

Sequence Grabber Channel Components

s	Contains a component instance that identifies the sequence grabber component that has connected to your channel component. The sequence grabber component provides this value to your channel component when it calls your <code>SGInitChannel</code> function (described on page 6-38).
c	Identifies the connection to your channel.
p	Specifies the location of the data to be added to the movie.
len	Contains the number of bytes of data to be added to the movie.
offset	Contains a pointer to a long integer that is to receive the offset to the new data in the movie. The sequence grabber component returns an offset that is correct in the context of a movie resource, even if the movie data is currently stored in memory. That is, if the movie is in memory, the returned offset reflects the location that the data will have in a movie on a permanent storage device, such as a disk.

DESCRIPTION

The `SGWriteMovieData` function behaves differently depending upon when you call it. If you call it from your `SGWriteSamples` function, this function writes the movie data to the device that contains the recording operation's movie file. If you call this function at other times, it may write the movie data to a movie in memory, depending upon the recording options that are in effect.

RESULT CODES

- File Manager errors
- Memory Manager errors

SGAddFrameReference

The `SGAddFrameReference` function allows your channel component to store sample references.

```
pascal ComponentResult SGAddFrameReference (SeqGrabComponent s,
                                             SeqGrabFrameInfo *frameInfo);
```

s	Contains a component instance that identifies the sequence grabber component that has connected to your channel component. The sequence grabber component provides this value to your channel component when it calls your <code>SGInitChannel</code> function (described on page 6-38).
frameInfo	Contains a pointer to a frame information structure (defined by the <code>SeqGrabFrameInfo</code> data type). Your component must completely specify the reference by placing the appropriate information into the record referred to by this parameter. The format and content of the frame information structure are described on page 6-84.

**DESCRIPTION**

The sequence grabber component uses the information you provide to create a new sample reference in the movie that contains the captured data. You supply the information for the reference in a frame information structure.

**RESULT CODES**

Memory Manager errors

**SEE ALSO**

Your component can retrieve these references by calling the `SGGetNextFrameReference` function, which is described in the next section.

## SGGetNextFrameReference

---

The `SGGetNextFrameReference` function allows your channel component to retrieve the sample references you stored by calling the `SGAddMovieData` or `SGAddFrameReference` function, described on page 6-85 and in the previous section, respectively.

```
pascal ComponentResult SGGetNextFrameReference
```

```
(SeqGrabComponent s,  
 SeqGrabFrameInfo *frameInfo,  
 TimeValue *frameDuration,  
 long *frameNumber);
```

- |                        |  |
|------------------------|--|
| <code>s</code>         | Contains a component instance that identifies the sequence grabber component that has connected to your channel component. The sequence grabber component provides this value to your channel component when it calls your <code>SGInitChannel</code> function (described on page 6-38).   |
| <code>frameInfo</code> | Contains a pointer to a frame information structure (defined by the <code>SeqGrabFrameInfo</code> data type), which is described on page 6-84. Your component must identify itself to the sequence grabber component by setting the <code>frameChannel</code> field of this structure to the component instance that identifies the current connection to your channel. The sequence grabber component then returns information about the specified frame in the remaining fields of this structure. |

## Sequence Grabber Channel Components

`frameDuration`

Contains a pointer to a time value. The sequence grabber component calculates the duration of the specified frame and returns that duration in the structure referred to by this parameter. Note that the sequence grabber component cannot calculate the duration of the last frame in a sequence. In this case, the sequence grabber component sets the returned time value to -1.

`frameNumber`

Contains a pointer to a long integer. Your channel component specifies the frame number corresponding to the frame about which you want to retrieve information. Frames are numbered starting at 0. However, frame numbers need not start at 0, and they may not be sequential. Set the field referred to by the `frameNumber` parameter to -1 to retrieve information about the first frame in a movie.

The sequence grabber component returns the frame number of the movie's next frame into the field referred to by this parameter. You can use this value the next time you call `SGGetNextFrameReference`.

**DESCRIPTION**

The `SGGetNextFrameReference` function allows your channel component to process these references sequentially or randomly—you specify the relative frame for which you want to retrieve information. The sequence grabber component then retrieves and returns information for that frame. Typically, your channel component calls this function within its `SGWriteSamples` function (described on page 6-43).

**RESULT CODE**

`paramErr`     -50     Invalid parameter specified

**SGSortDeviceList**

The `SGSortDeviceList` function allows you to sort your device list alphabetically.

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

`s`                Specifies the component instance that identifies the sequence grabber component that is using your channel. The sequence grabber provides this to you when it calls your `SGInitChannel` function (described on page 6-38).

`list`            Contains a pointer to a device list structure pointer.

## Sequence Grabber Channel Components

## DESCRIPTION

Your component constructs its device list whenever the sequence grabber calls your `SGGetChannelDeviceList` function (described on page 6-60). You may add entries to the device list in any order you like. Once you have built up your device list, you may use the `SGSortDeviceList` function to sort that list alphabetically, by device name. The sequence grabber correctly updates the `selectedIndex` field in the device list structure, as well.

The format and content of the device list structure are discussed earlier in this chapter, in “Working With Channel Devices” beginning on page 6-58.

## RESULT CODE

`paramErr`     -50     Invalid parameter value

## SGChangedSource

---

The `SGChangedSource` function allows you to tell the sequence grabber that your component is now using a different device.

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

<code>s</code>	Specifies the component instance that identifies the sequence grabber component that is using your channel. The sequence grabber provides this to you when it calls your <code>SGInitChannel</code> function (described on page 6-38).
<code>c</code>	Identifies the connection to your channel.

## DESCRIPTION

Applications can instruct your channel to change its input device, for example, by calling the sequence grabber's `SGSetChannelDevice` function. The sequence grabber passes this request on to your channel component. Whenever you successfully change your input device, you should tell the sequence grabber by calling its `SGChangedSource` function. This allows the sequence grabber to update the information it keeps about your channel.



## Summary of Sequence Grabber Channel Components

---

### C Summary

---

#### Constants

---

```

/* sequence grabber channel component type */
#define SeqGrabChannelType 'sgch'

/* 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 */
};

/* 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 */

/* 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 */
};

```

## Sequence Grabber Channel Components

```

};
typedef unsigned char SeqGrabChannelInfoEnum;

/* basic sequence grabber channel component selectors */
kSGSetGWorldSelect          = 0x4;   /* SetGWorld */
kSGStartPreviewSelect       = 0x10;  /* SGStartPreview */
kSGStartRecordSelect        = 0x11;  /* SGStartRecord */
kSGIdleSelect               = 0x12;  /* SGIdle */
kSGStopSelect               = 0x13;  /* SGStop */
kSGPauseSelect              = 0x14;  /* SGPause */
kSGPrepareSelect             = 0x15;  /* SGPrepare */
kSGReleaseSelect            = 0x16;  /* SGRelease */
kSGUpdateSelect             = 0x27;  /* SGUpdate */

/* selectors for common channel configuration functions */
kSGCSetChannelUsageSelect    = 0x80;  /* SGSetChannelUsage */
kSGCGetChannelUsageSelect    = 0x81;  /* SGGetChannelUsage */
kSGCSetChannelBoundsSelect   = 0x82;  /* SGSetChannelBounds */
kSGCGetChannelBoundsSelect   = 0x83;  /* SGGetChannelBounds */
kSGCSetChannelVolumeSelect   = 0x84;  /* SGSetChannelVolume */
kSGCGetChannelVolumeSelect   = 0x85;  /* SGGetChannelVolume */
kSGCGetChannelInfoSelect     = 0x86;  /* SGGetChannelInfo */
kSGCSetChannelPlayFlagsSelect = 0x87;  /* SGSetChannelPlayFlags */
kSGCGetChannelPlayFlagsSelect = 0x88;  /* SGGetChannelPlayFlags */
kSGCSetChannelMaxFramesSelect = 0x89;  /* SGSetChannelMaxFrames */
kSGCGetChannelMaxFramesSelect = 0x8A;  /* SGGetChannelMaxFrames */
kSGCSetChannelRefConSelect    = 0x8B;  /* SGSetChannelRefCon */
kSGCSetChannelClipSelect     = 0x8C;  /* SGSetChannelClip */
kSGCGetChannelClipSelect     = 0x8D;  /* SGGetChannelClip */
kSGCGetChannelSampleDescriptionSelect = 0x8E;
                                   /* SGGetChannelSampleDescription */
kSGCGetChannelDeviceListSelect = 0x8F;  /* SGGetChannelDeviceList */
kSGCSetChannelDeviceSelect    = 0x90;  /* SGSetChannelDevice */
kSGCSetChannelMatrixSelect    = 0x91;  /* SGSetChannelMatrix */
kSGCGetChannelMatrixSelect    = 0x92;  /* SGGetChannelMatrix */
kSGCGetChannelTimeScaleSelect = 0x93;  /* SGGetChannelTimeScale */

/* selectors for video channel configuration functions */
kSGCGetSrcVideoBoundsSelect   = 0x100; /* SGGetSrcVideoBounds */
kSGCSetVideoRectSelect        = 0x101; /* SGSetVideoRect */
kSGCGetVideoRectSelect        = 0x102; /* SGGetVideoRect */
kSGCGetVideoCompressorTypeSelect = 0x103; /* SGGetVideoCompressorType */

```

## Sequence Grabber Channel Components

```

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 */
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 channel control functions */
kSGCInitChannelSelect               = 0x180; /* SGCInitChannel */
kSGCWriteSamplesSelect              = 0x181; /* SGCWriteSamples */

```

## Sequence Grabber Channel Components

```

kSGCGetDataRateSelect          = 0x182; /* SGDataRate */
kSGCAlignChannelRectSelect     = 0x183; /* SAlignChannelRect */
};

/* values for pause parameter of SGPause function */
enum {
    seqGrabUnpause = 0, /* restart the current operation */
    seqGrabPause   = 1, /* pause the current operation */
};

```

## Data Types

---

```

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 */
};

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;

```

## Functions

---

### Configuring Sequence Grabber Channel Components

```

pascal ComponentResult SGInitChannel
    (SGChannel c, SeqGrabComponent owner);

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

```

**Controlling Sequence Grabber Channel Components**

```

pascal ComponentResult SGStartPreview
    (SeqGrabComponent s);
pascal ComponentResult SGStartRecord
    (SeqGrabComponent s);
pascal ComponentResult SGIdle
    (SeqGrabComponent s);
pascal ComponentResult SGUpdate
    (SeqGrabComponent s, RgnHandle updateRgn);
pascal ComponentResult SGStop
    (SeqGrabComponent s);
pascal ComponentResult SGWriteSamples
    (SGChannel c, Movie m, AliasHandle theFile);
pascal ComponentResult SGPause
    (SeqGrabComponent s, Byte pause);
pascal ComponentResult SGPrepare
    (SeqGrabComponent s, Boolean prepareForPreview,
     Boolean prepareForRecord);
pascal ComponentResult SGRelease
    (SeqGrabComponent s);

```

**Configuration Functions for All Channel Components**

```

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 SGSetChannelRefCon
    (SGChannel c, long refCon);
pascal ComponentResult SGGetDataRate
    (SGChannel c, long *bytesPerSecond);
pascal ComponentResult SGGetChannelSampleDescription
    (SGChannel c, Handle sampleDesc);

```

## Sequence Grabber Channel Components

```

pascal ComponentResult SGGetChannelTimeScale
    (SGChannel c, TimeScale *scale);
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 SGSetChannelDevice
    (SGChannel c, StringPtr name);

```

**Configuration Functions for Video Channel Components**

```

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

```

## Sequence Grabber Channel Components

```

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

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);

pascal ComponentResult SGAlignChannelRect
    (SGChannel c, Rect *r);

```

**Configuration Functions for Sound Channel Components**

```

pascal ComponentResult SGSetChannelVolume
    (SGChannel c, short volume);

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

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);

```

## Sequence Grabber Channel Components

```

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);

```

**Utility Functions for Sequence Grabber Channel Components**

```

pascal ComponentResult SGAddMovieData
    (SeqGrabComponent s, SGChannel c, Ptr p,
     long len, long *offset, long chRefCon,
     TimeValue time, short writeType);

pascal ComponentResult SGWriteMovieData
    (SeqGrabComponent s, SGChannel c,
     Ptr p, long len, long *offset);

pascal ComponentResult SGAddFrameReference
    (SeqGrabComponent s,
     SeqGrabFrameInfo *frameInfo);

pascal ComponentResult SGGetNextFrameReference
    (SeqGrabComponent s,
     SeqGrabFrameInfo *frameInfo,
     TimeValue *frameDuration,
     long *frameNumber);

pascal ComponentResult SGSortDeviceList
    (SeqGrabComponent s, SGDeviceList list);

pascal ComponentResult SGChangedSource
    (SeqGrabComponent s, SGChannel c);

```



## Pascal Summary

---

### Constants

---

CONST

```
SeqGrabChannelType = 'sgch'; {sequence grabber channel component type}

{device list structure flags}
sgDeviceListWithIcons           = 1;      {include icons}
sgDeviceListDontCheckAvailability = 2;      {don't check available }
                                         { device list}

{flags for SGSetChannelPlayFlags and SGGetChannelPlayFlags functions}
channelPlayNormal               = 0      {use default play methodology}
channelPlayFast                 = 1;      {sacrifice playback quality }
                                         { for specified rate}
channelPlayHighQuality          = 2;      {sacrifice playback rate }
                                         { for image quality}
channelPlayAllData              = 4;      {play all captured data }
                                         { including that stored in }
                                         { offscreen buffers}

{flags for SGSetChannelUsage and SGGetChannelUsage functions}
seqGrabRecord                   = 1;      {used during record operations}
seqGrabPreview                  = 2;      {used during preview operations}
seqGrabPlayDuringRecord        = 4;      {used during record operations}

{SGGetChannelInfo function flags}
seqGrabHasBounds                = 1;      {visual representation of data}
seqGrabHasVolume                = 2;      {audio representation of data}
seqGrabHasDiscreteSamples       = 4;      {data organized in discrete frames}

{basic sequence grabber channel component selectors}
kSGSetGWorldSelect              = $4;      {SGSetGWorld}
kSGStartPreviewSelect           = $10;     {SGStartPreview}
kSGStartRecordSelect            = $11;     {SGStartRecord}
kSGIdleSelect                   = $12;     {SGIdle}
kSGStopSelect                   = $13;     {SGStop}
kSGPauseSelect                  = $14;     {SGPause}
kSGPrepareSelect                = $15;     {SGPrepare}
```

## Sequence Grabber Channel Components

```

kSGReleaseSelect          = $16;    {SGRelease}
kSGUpdateSelect           = $27;    {SGUpdate}

{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;    {SGSetChannelClip}
kSGCGetChannelClipSelect   = $8D;    {SGGetChannelClip}
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}
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}

```

## Sequence Grabber Channel Components

```

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 channel control functions}
kSGCInitChannelSelect        = $180; {SGCInitChannel}
kSGCWriteSamplesSelect       = $181; {SGCWriteSamples}
kSGCGetDataRateSelect        = $182; {SGCDataRate}

{values for the pause parameter of the SGPause function}
seqGrabUnpause    = 0;    {restart current operation}
seqGrabPause      = 1;    {pause the current operation}

```

## Data Types

---

### TYPE

```
SeqGrabFrameInfo =
```

### RECORD

```

    frameOffset:    LongInt;    {offset to the sample}
    frameTime:      LongInt;    {time that frame was captured}

```

## Sequence Grabber Channel Components

```

    frameChannel:      SGChannel;    {current connection to channel}
    frameRefCon:       LongInt;      {reference constant for channel}
END;

SGDeviceListPtr = ^SGDeviceListRecord;
SGDeviceList = ^SGDeviceListPtr;

SGDeviceListRecord =
RECORD
    count:             Integer;      {count of devices}
    selectedIndex:     Integer;      {current device}
    reserved:          LongInt;      {set to 0}
    entry:             ARRAY[0..] OF SGDeviceName; {device names}
END;

SGDeviceName =
RECORD
    name:              Str63;        {device name}
    icon:              Handle;        {device icon}
    flags:             LongInt;      {flags}
    refCon:            LongInt;      {set to 0}
    reserved:          LongInt;      {set to 0}
} END;

```

## Routines

**Configuring Sequence Grabber Channel Components**

```

FUNCTION SGInitChannel      (c: SGChannel; owner: SeqGrabComponent):
                             ComponentResult;

FUNCTION SGSetGWorld        (s: SeqGrabComponent; gp: CGrafPtr;
                             gd: GDHandle): ComponentResult;

```

**Controlling Sequence Grabber Channel 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 SGWriteSamples      (c: SGChannel; m: Movie; theFile: AliasHandle):
                             ComponentResult;

```

## Sequence Grabber Channel Components

```

FUNCTION SGPause          (s: SeqGrabComponent; pause: Byte):
                           ComponentResult;

FUNCTION SGPrepare        (s: SeqGrabComponent;
                           prepareForPreview: Boolean;
                           prepareForRecord: Boolean): ComponentResult;

FUNCTION SGRelease        (s: SeqGrabComponent): ComponentResult;

```

**Configuration Routines for All Channel Components**

```

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;

FUNCTION SGGetChannelMaxFrames
                           (c: SGChannel; VAR frameCount: LongInt):
                           ComponentResult;

FUNCTION SGSetChannelRefCon
                           (c: SGChannel; refCon: LongInt):
                           ComponentResult;

FUNCTION SGGetDataRate     (c: SGChannel;
                           VAR bytesPerSecond: LongInt): ComponentResult;

FUNCTION SGGetChannelSampleDescription
                           (c: SGChannel; sampleDesc: Handle):
                           ComponentResult;

FUNCTION SGGetChannelTimeScale
                           (c: SGChannel; VAR scale: TimeScale):
                           ComponentResult;

FUNCTION SGSetChannelClip  (c: SGChannel; theClip: RgnHandle):
                           ComponentResult;

FUNCTION SGGetChannelClip  (c: SGChannel; VAR theClip: RgnHandle):
                           ComponentResult;

```

## Sequence Grabber Channel Components

```

FUNCTION SGSetChannelMatrix
    (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 SGSetChannelDevice
    (c: SGChannel; name: StringPtr):
        ComponentResult;

```

**Configuration Routines for Video Channel Components**

```

FUNCTION SGSetChannelBounds (c: SGChannel; bounds: Rect): ComponentResult;
FUNCTION SGGetChannelBounds (c: SGChannel; VAR bounds: Rect):
    ComponentResult;

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;

```

## Sequence Grabber Channel Components

```

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;

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;

FUNCTION SGAlignChannelRect
    (c: SGChannel; VAR r: Rect): ComponentResult;

```

**Configuration Routines for Sound Channel Components**

```

FUNCTION SGSetChannelVolume
    (c: SGChannel; volume: Integer):
        ComponentResult;

FUNCTION SGGetChannelVolume
    (c: SGChannel; VAR volume: Integer):
        ComponentResult;

FUNCTION SGSetSoundInputDriver
    (c: SGChannel; driverName: Str255):
        ComponentResult;

FUNCTION SGGetSoundInputDriver
    (c: SGChannel): LongInt;

FUNCTION SGSoundInputDriverChanged
    (c: SGChannel): ComponentResult;

FUNCTION SGSetSoundRecordChunkSize
    (c: SGChannel; seconds: LongInt):
        ComponentResult;

```

## Sequence Grabber Channel Components

```

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;

```

**Utility Routines for Sequence Grabber Channel Components**

```

FUNCTION SGAddMovieData    (s: SeqGrabComponent; c: SGChannel; p: Ptr;
    len: LongInt; VAR offset: LongInt;
    chRefCon: LongInt; time: TimeValue;
    writeType: Integer): ComponentResult;

FUNCTION SGWriteMovieData  (s: SeqGrabComponent; c: SGChannel; p: Ptr;
    len: LongInt; VAR offset: LongInt):
    ComponentResult;

FUNCTION SGAddFrameReference
    (s: SeqGrabComponent;
     VAR frameInfo: SeqGrabFrameInfo):
    ComponentResult;

FUNCTION SGGetNextFrameReference
    (s: SeqGrabComponent;
     VAR frameInfo: SeqGrabFrameInfo;
     VAR frameDuration: TimeValue;
     VAR frameNumber: LongInt): ComponentResult;

FUNCTION SGSortDeviceList  (s: SeqGrabComponent; list: SGDeviceList):
    ComponentResult;

FUNCTION SGChangedSource   (s: SeqGrabComponent; c: SGChannel):
    ComponentResult;

```



Result Codes

noDeviceForChannel	-9400	Channel component cannot find its device
cantDoThatInCurrentMode	-9402	Request invalid in current mode
notEnoughMemoryToGrab	-9403	Insufficient memory for record operation
notEnoughDiskSpaceToGrab	-9404	Insufficient disk space for record operation
seqGrabInfoNotAvailable	-9407	Channel component cannot support request
deviceCantMeetRequest	-9408	Device cannot support grabber

