Oct. 25, 1994

Dear OM1 member:

Some changes in the final document:

- Encoding recommendations are different, to reflect Bernard Szabo's latest letter.
- Added language to indicate timing of functions' return.
- A hardware error is assumed to be fatal.
- Added language to clarify the use of OM1M_PRIV.
- OM1_SET cannot set a driver parameter, only a stream parameter.
- Removed the various "after end" modes in OM1_STEP, as it's only reasonable to go into the pause mode after a step.
- Clarified OM1I_UPD_PALETTE.
- Added OM1I_UPD_VGA_MODE, as in Sigma's documentation.
- Defined some symbols as OEM instead of RESERVED, for each vendor's use. Note that compilers will probably flag the use of the same name for multiple symbols.

As usual, my contact info is:

phone: +1 408 428 6600 ext 6767
fax: +1 408 428 9871
e-mail: lcheng@creaf.com

Regards,

Lawrence Cheng

Open MPEG (OM1) DOS API

version 1.00p0.66

DRAFT - October 2519, 1994

M

Introduction

The purpose of the Open MPEG (OM1) consortium is to promote the use of MPEG in the consumer market. One way to accomplish this is by specifying a common Applications Programming Interface (API). This API is used by various applications to control and communicate, in a uniform manner, with different vendors' MPEG hardware.

This document describes the OM1 API version 0.65, which may be used to interface an MPEG decoder/display board to applications running under MS-DOS[tm].

This API may be freely used by anyone to develop MPEG related products and is provided without licensing fees or royalties of any kind. It represents the effort of interested manufacturers, software developers, and content developers to meet the need for a widely available and public MPEG API. It is based on proposals and comments presented to the committee. Interested companies are invited to join this committee to participate in future enhancements of the API as well as future OM1 projects.

The specification presented here is provided without warranty or guarantee of usability or merchandisability. Use of this specification does not imply licensing of intellectual property associated to ISO 11172 (MPEG-1) or any derivation of that standard.

Features and overview

This document describes an MPEG API for DOS. It is intended to provide the basic capabilities to play back MPEG streams on a wide variety of hardware.

The API addresses the needs of both simple and sophisticated applications. It includes file handling functions, so that an application can play an MPEG stream simply by opening a file. It may also be used to implement more complex systems where:

- multiple streams are pieced together according to an event and played in real time. An example might be a game where the hero is running down a corridor, and when he gets to the end of the corridor a selection is made in real time to go left or right. The appropriate MPEG stream is played in response to the selection. The overall effect is such that the video appears to be continuous.
- private data from the MPEG stream is passed back to the application. An example might be VGA bitmaps which are encoded within the MPEG stream. The overlay graphics are encoded by the application developer such that when the MPEG stream is played the overlay graphics (or alpha channel) are presented in time to update the screen.
- MPEG stream parameters are passed back to the application. An example might be the implementation of a step or seek function where the first approximation of the stream position is computed from the encoded bit rate of the stream.

There are also facilities for reading the capabilities of the hardware, for setting the audio and video parameters of the display, and for setting callback functions that can be associated either with a particular stream or with the driver in general.

Usage notes

Installation

The driver must be loaded before the application starts. It installs itself at a free interrupt from 80h to FFh. The application can identify it by searching the string "MPEGVIDEO" at the address pointed to by the interrupt vector plus 13 in Pascal and 14 in C. In fact, the string is <13,MPEGVIDEO,0>;so at offset 13, the application can find a Pascal-type string and at offset 14 a C-type string.

To install the driver correctly, the application should follow this procedure. First, search for the driver by scanning interrupt vectors. If found, record the interrupt number. If it is not found, the driver was not loaded before launching the application. It can either quit with an error or programmatically load the driver.

If your application loads the driver, it must send the command OM1_UNLOAD to the driver before it closes

The application can then get information about the driver (name, version) and the state of the hardware with the OM1_GET command.

Streams

The only types of streams allowed are system multiplexed MPEG streams that contain 0 to 15 video streams and 0 to 31 audio streams. Streams are provided in two ways:

- from a file. The driver directly processes the file format and buffers.
- from a buffer. The calling application provides the stream data. The application can pull a scrambled stream from another location, descramble it, then present the stream in standard MPEG format to the decoder. The only buffers the OM1 API can access are those located in low memory, so it is up to the application to copy any data from upper or extended memory into low memory.

NOTE: For convention, we assume in this document that the term “stream(s),” without a qualifier, means “system multiplexed bitstream(s).” Otherwise, the qualifier “video,” “audio,” “private1,” “private2,” “padding,” or “elementary” should precede the term “stream.”

Stream modes

A stream can be in one of these modes:

- stop: No display for video, no output for audio.
- pause: Audio is muted. The last picture is frozen in the video window.
- play: Stream is playing; audio and video are active.
- seek: Stream is reaching a given position. Audio is muted, video is either blank or frozen.
- step: Stream is in the process of stepping to the next specified video picture. Audio is muted.
- frozen: The last picture is frozen in the video window while audio is playing.

The ready modes are the pause and the stop modes. Note that the modes are not necessarily related to the result of the commands bearing the same name - for example, the OM1_SEEK may set

the stream in pause or stop or play mode upon completion.

Stream counter

The stream counter counts the stream bytes as they enter the system target decoder (STD). The counter is stopped in the pause and stop modes, counting in the play, step, and frozen modes, and is being set in the seek mode. The stream counter's value is reflected in OM1I_STM_POSITION.

Source and destination windows

The frame buffer is the area of memory which contains the entire decompressed picture. The source video window is the rectangular portion of the frame buffer that is displayed. The destination video window is the rectangular portion of the display where the picture is placed. The origins of both frame buffer and display are in the upper left-hand corner.

The parameters OM1I_VID_SRC_SIZE and OM1I_VID_DEST_SIZE allow the picture to be cropped and scaled independently in the horizontal and vertical directions.

Keying modes

There are several ways to control the way graphics pixels are replaced by video pixels in the destination window. The mode is set by OM1I_VID_KEY_MODE.

OM1F_ALL_VGA	All the graphics pixels are displayed in the destination window. Equivalent to hiding the video.
OM1F_ALL_VID	All the video pixels are displayed in the destination window. Equivalent to hiding the graphics.
OM1F_KEY_VGA (default)	Key on VGA, or color key. All graphics pixels which match the key color, after the key mask is applied to it, are transparent and replaced by video pixels. The key color is an index in the palette or an RGB color, depending on the VGA mode. The key mask allows a range of colors to be selected as a color key; in effect, if (VGAPixelColor & !KeyMask) == (VGAKeyColor & !KeyMask) show video pixel A key mask of 0 has no effect. Key mask support is optional, and is determined by reading the driver capabilities.
OM1F_KEY_VID	Key on video. All pixels in the video destination window are compared against minimum and maximum RGB888 or YCbCr key colors. If the pixels are within this range, they are not shown, and are replaced by graphics pixels. In effect, if (VideoPixelColor >= OM1I_VID_KEY_MIN) && (VideoPixelColor <= OM1I_VID_KEY_MAX) show VGA pixel

	Key on video support is optional, and is determined by reading the driver capabilities.
OM1F_KEY_MIX	A combination of color key and key on video. If both the key on VGA and key on video tests pass, then the VGA key color is displayed.

Memory usage

The only restriction is that buffers which are used to communicate with applications must reside in low memory.

Opening a file stream

This example illustrates how to initialize the driver, open a file stream, and play it.

```
// Callback function - only pseudocode here.  Look below for an example.
WORD far _loadds CallbackFct(BYTE Message,BYTE hStream,DWORD Value)
{
        .
        .
        .
}

// Error function - Writes Msg and stops the program.
void Error(char *Msg,int ExitCode)
{
        fprintf(stderr,Msg);
        exit(ExitCode);
}

void main(int argc,char *argv[])
{
        BYTE hStream;

        if (argc<2)
            Error("Specify a file to play.\n",1);

        // Locate the driver.
        if (!FindDriver())
            Error("Driver not found.\n",2);

        // Re-init the driver, as other applications may have changed values.
        OM1Init();

        // Install the callback function for the driver, i.e. handle of 0.
```

```
        OM1Callback(0,(DWORD)CallbackFct);

        OM1Set(0,OM1I_VID_DEST_SIZE,MAKEDWORD(352,240));
        OM1Set(0,OM1I_VID_DEST_POS ,MAKEDWORD(174,80));

        // Open the file.
        hStream=OM1Open(OM1F_FILE,(DWORD)(LPSTR)argv[1]);

        // If hStream is null, the file has not been properly opened.
        if (!hStream)
            Error("Error while opening the file.\n",3);



        // Play the file.
        OM1Play(hStream,OM1F_END_PAUSE,0);

        // Wait loop
        // exits when the stream is stopped or a key is pressed.
        while (!kbhit()&&!(OM1Get(hStream,OM1I_STM_MODE)&OM1F_READY));

        // Close the stream.
        OM1Close(hStream);
}
```

Opening a buffered stream

This example illustrates how to initialize the driver, open a buffered stream, and play it.

```
// Buffers should not be too big
// to avoid long DOS access.
#define BUF_SIZE 5000

// We use a structure to store the buffers information
// in the USER field of the associated stream.
// We use a ping pong buffer.
struct TBuf
{
        FILE *f;
        WORD Size;
        int BufNb;
        char *Buffers[2];
};

// The callback function.
// Don’t forget the <far _loadds> or <huge> attributes.
// Prefer using DOS open, read and seek functions for best performance;
// here, we use standard C functions for compatibility.
WORD far _loadds CallbackFct(BYTE Message,BYTE hStream,DWORD Value)
{
```

```
// Get our buffer structure address in the USER field of the stream.
// This is not useful for the OM1F_BUF_CREATE.
struct TBuf *Buf=(struct
TBuf*)OM1Command(OM1_GET,hStream,OM1I_STM_USER,0L);
switch (Message)
{
	// First message received - make all your allocations here.
	case OM1M_BUF_CREATE :
		Buf=(struct TBuf*)malloc(sizeof(struct TBuf));
		// Value contains the value passed when opening the file;
		// here, it's the filename.
		Buf->f=fopen((char*)Value,"rb");
		// If we cannot open the file, return an error.
		// Note : the OM1M_BUF_CLOSE is not called when
		// an error occurs during the creation.
		if (!Buf->f)
		{
			free(Buf);
			return OM1E_DOS;
		}
		// Allocate our 2 buffers.
		Buf->Buffers[0]=(BYTE *)malloc(BUF_SIZE);
		Buf->Buffers[1]=(BYTE *)malloc(BUF_SIZE);;
		Buf->BufNb=0;
		// Store the structure address in the USER field.
		OM1Set(hStream,OM1I_STM_USER,(DWORD)(BYTE far *)Buf);
		// We want to prepare the next buffer when at least 1 byte
		// of the other one has been read.
		OM1Set(hStream,OM1I_BUF_POS,1);
		break;

	// Message received when closing the stream - delete buffers.
	case OM1M_BUF_CLOSE :
		fclose(Buf->f);
		free(Buf->Buffers[0]);
		free(Buf->Buffers[1]);
		free(Buf);
		break;

	// When receiving this one, you have to seek to the position
	// in Value (in bytes).
	case OM1M_BUF_SEEK :
		fseek(Buf->f,Value,SEEK_SET);
		break;

	// Message received when a buffer has reached its signal position;
	// prepare here the next buffer.
	case OM1M_BUF_POS  :
```

```
                        Buf->Size=fread(Buf->Buffers[Buf->BufNb],1,BUF_SIZE,Buf->f);
                        break;

                // Message indicating a buffer has been completely read.
                // Switch to the other one.
                case OM1M_BUF_EMPTY:
                        OM1Set(hStream,OM1I_BUF_SIZE,Buf->Size);
                        OM1Set(hStream,OM1I_BUF_OFFSET,LOWORD(Buf->Buffers[Buf-
                                >BufNb]));
                        OM1Set(hStream,OM1I_BUF_HANDLE,OM1F_BUF_LOW|
HIWORD((BYTE far                *)Buf->Buffers[Buf->BufNb]));
                        Buf->BufNb++;
                        Buf->BufNb%=2;
                        break;
        }
        return 0;
}

// Error function - writes Msg and stop the program.
void Error(char *Msg,int ExitCode)
{
        fprintf(stderr,Msg);
        exit(ExitCode);
}

void main(int argc,char *argv[])
{
        BYTE hStream;
        DWORD d;

        if (argc<2)
                Error("Specify a file to play.\n",1);

        // Locate the driver.
        if (!FindDriver())
                Error("Driver not found.\n",2);

        // Re-init the driver.
        OM1Init();

        // Install the callback function.
        // C type declared in the macro OM1Callback.
        // Declared as a driver callback (0 handle).
        // If you want the callback to apply only to a stream, copy this line
        // after the OM1Open and specify hStream instead of 0.
        OM1Callback(0,(DWORD)CallbackFct);

        // Open the file with the OM1F_BUFFERS flag to indicate we
```

```
        // will provide data. The value parameter is passed to the callback
        // function. Here, we give the filename.
        hStream=OM1Open(OM1F_BUFFERS,(DWORD)(LPSTR)argv[1]);

        // If hStream is null, the file has not been properly opened.
        if (!hStream)
                Error("Error while opening the file.\n",3);



        // Set the size of the destination window.
        OM1Set(hStream,OM1I_VID_DEST_SIZE,MAKEDWORD(352,240));
        OM1Set(hStream,OM1I_VID_DEST_POS ,MAKEDWORD(174,80));

        // Play the file.
        OM1Play(hStream,OM1F_END_PAUSE,0);

        // Wait loop - exits when the stream is stopped or a key is pressed.
        while (!kbhit()&&!(OM1Get(hStream,OM1I_STM_MODE)&OM1F_READY));

        // Close the stream.
        OM1Close(hStream);
}
```

Encoding and hardware recommendations

To ensure compatibility with a wide range of hardware, the committee recommends the following:

VGA display

Shared frame buffers generally have difficulty dealing with a palette of less than 256 colors. Similarly many overlay processors have difficulty with 24-bit true-color modes. This specification recommends that a palette of 256 colors, a 32k colors, or 64k colors are used; the application should not use any text modes.

VBE 2.0

Some devices may not be VGA-compatible. If the device supports VESA BIOS Extensions 2.0, then applications are strongly recommended to use VBE calls instead of OM1_UPDATE to load palette data or to set graphics mode.

Timeliness of display

Hardware operating under the OM1 specification need to be capable of piecing together video sequences in real time. This means that internal driver buffers and control logic must be designed to minimize delays between the time data is presented to the OM1 driver and the time audio and video is displayed

Color/gamma correction

For many applications it is desirable to display video within the context of a VGA display. Ideally video pictures that are captured using OM1_CAPTURE are gamma adjused to match the typical characteristics of the VGA display.

Encoding MPEG Stream

The following is brief summary of the encoder group's conclusions. Please refer to the entire encoding group recommendation. At the time of this writing the recommendations are:

- Encoders which alter quantizer matrices should precede each MPEG GOP with a sequence header. Other encoders - that is, those which use fixed quantizer matrices - should generate only one sequence header at the beginning of a stream.
- Audio PTS fields should be included near video entry points to enable decoders to rapidly commence random access playback.
- Encoders should generate streams which fully comply with MPEG (ISO11172) syntax and semantics.

1. Encoded streams consist of a system stream that incorporates a video stream, one or more audio streams, and optionally private data streams.
2. An MPEG stream must begin with a *sequence header*. Optionally, if a sequence header is provided later in the stream, it must be provided before each *Group Of Pictures (GOP).*
3. The information necessary to insure synchronization between the video and audio portions of the MPEG stream is contained in the system stream layer. To maintain synchronization it is necessary to have an *Audio Presentation Time Stamp (PTS)* along with a *Video Presentation*

Time Stamp (PTS) with the same value. This provides a *synchronization point* the decoder and/or application can use as a reference into the stream. *Synchronization points must occur at least once every 0.7 seconds*.

4. *Groups of Pictures (GOP)* provide a convenient method to meter events associated with the MPEG stream. The beginning of a GOP may be used as a call back condition. Thus the MPEG may be encoded to break up video sequences so the GOPs occur when the position of the mouse or other event must be checked.
5. It is recommended that every GOP begins with a *synchronization point.*

Command Set

Here is a summary of the commands classed by function.

- To initialize the driver

OM1_INIT initializes the driver

· To open and close streams

OM1_OPEN opens a stream
OM1_CLOSE closes a stream

- To play the stream

OM1_PLAY plays a stream
OM1_PAUSE pauses a stream
OM1_FREEZE freezes the displayed picture, audio continues playing
OM1_STOP stops a stream
OM1_SEEK seek to a position in a stream
OM1_STEP step pictures for video streams

- To manage group streams

OM1_GROUP selects audio & video streams within a system stream

- To set and get parameters about the driver and streams

OM1_SET sets a parameter
OM1_GET gets a parameter

- To do specials functions on streams

OM1_CALLBACK installs a callback function for a stream
OM1_SIGNAL installs signals at defined positions or times in a stream

- To optionally capture a picture from a paused stream

OM1_CAPTURE captures the currently displayed picture of a paused stream

- To send special hardware commands

OM1_UPDATE
OM1_COPY_TO_OVERLAY

- To unload the driver

OM1_UNLOAD

The parameters are generally of the form :
BH : command id
BL : stream handle
CX : flags (eventually combined with a value)
DX,AX : a 32 bit value (high word in DX) or a pointer with the segment in DX and the offset in AX.

In return, BX is zero if the command is successful; otherwise it indicates the error code in BH. If the error code indicates a DOS error, BL contains the DOS error code. Generally, the driver returns with the required value or error, and this completes the execution of the command. However, with OM1_PLAY, OM1_SEEK, and OM1_STEP, the driver should return after examining the validity of the parameters. For these three, other mechanisms exist to report error in execution (e.g. OM1E_ERROR), completion (e.g. OM1M_COMPLETED), or mode (e.g. OM1I_STM_MODE).

If the command returns a value, it is always in DX,AX (high word in DX).

Only registers AX,BX,CX,DX are modified by the driver call.

OM1_CALLBACK

OM1_CALLBACK allows the application to install callback functions. These functions are called when a command is completed, when an error occurs, or when the driver needs data for a buffered stream. If the application specifies a zero handle, the callback applies to the driver and to all subsequently opened streams not associated with a specific callback function. The application can specify a null pointer if it doesn't want a callback function for a handle.

The application must install a callback for the driver immediately after the driver is initialized. If not, functions which use a callback but cannot find one will issue the error message OM1E_NO_CALLBACK.

Application developers should note that the driver can issue a callback within an interrupt handling routine. Furthermore, applications must not issue any commands to the driver while executing the callback function invoked by the driver.

Parameters

BH ⇨ OM1_CALLBACK
BL ⇨ handle of the stream; a zero handle specifies the global callback for the driver
CX ⇨ a flag specifying the type of call
OM1F_PASCAL ⇨ Pascal calling convention
OM1F_C ⇨ C calling convention
else values are passed in registers.
DX:AX ⇨ far pointer to the callback function

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
the handle of the stream is not valid.
OM1E_INVALID_FLAGS
Flags are invalid or incoherent
OM1E_INVALID_CMD
Unrecognized command code.

Notes

Depending on the flags the application specified when declaring its function, the values for the function are passed in registers or on the stack following the PASCAL convention or the C convention.

When the callback function is called, the value of DS may not be the application's DS; therefore, the function must load DS. In C, use the 'huge' or '_loadds' attributes. In Pascal, use an inline instruction or inline assembler to reload DS.

If the application uses the C or PASCAL convention, the function should look like this:

- in C :

unsigned short far _loadds MyCallback(unsigned char Message, unsigned char hStream, unsigned long Value);
or unsigned short huge MyCallback(...
or unsigned short far pascal _loadds MyCallback(... if declared with the OM1F_PASCAL flag

- in Turbo Pascal :

```
{$F+} (* if not already set *)
function MyCallback( Message , hStream : byte; Value : longint):word;
...
(* you can turn off far calls with {$F-} *)
```

in Turbo Pascal, one can use the following instructions at the beginning of a function to reload DS:

```
asm
    mov ax,SEG @Data
    mov ds,ax
end;
```

Driver parameters on callback to the application are:

- Message = BH ⇨ message id
- hStream = BL ⇨ handle of the stream that the message is concerned with
- Value = DX,AX

Application's return value to driver:

- Value = AX ⇨ returns zero if successful

Callback messages

Messages passed back to the application comprise:

OM1M_ERROR
Specifies that an error occurred while executing a command.
Value is the error code (AX), which may be one of these:

OM1E_DOS
A DOS error occurred while reading the stream. AL contains the DOS error code.
OM1E_HARDWARE
A fatal problem occurred with the hardware. AL contains information about the error.
OM1E_STREAM
The stream contains invalid data.

OM1M_COMPLETED
Issued when a command has completed. The stream is in a ready mode (paused or stopped).
Value is the completed command ID.

OM1M_CANCELED
Issued when a new command is sent before a previous one has completed.
Value is the canceled command ID.

OM1M_BUF_CREATE
Sent while opening a buffered stream. The application should allocate the buffers and initialize everything when receiving this message.
Value is the value passed in the OM1_OPEN command. (The application can use this to get a filename)
The application should return a zero if successful.

OM1M_BUF_CLOSE
Sent while closing a stream. It is the point where the application can release the memory that was allocated.
No Value.

OM1M_BUF_SEEK
Sent to ask the application to seek to a given position. This message is sent only during OM1_OPEN in order for the driver to determine what type of stream the file is. Because these data sometimes make up the first 50k of an MPEG file, the application may receive several OM1M_BUF_SEEK messages before OM1_OPEN completes.
Value is the position to reach in bytes.

OM1M_BUF_EMPTY
Sent when a buffered stream's buffer is empty and more data is needed to complete the actual command. The application can specify a new address on a new buffer with the OM1_SET-OM1I_BUF_HANDLE and OM1_SET-OM1I_BUFF_OFFSET command. If it doesn't, the current buffer will be scanned again if OM1F_BUF_LOOP is set; otherwise the stream is stopped.

Value is the current position of the stream.

OM1M_BUF_POS
Sent when a buffered stream's buffer has reached the position specified with OM1I_BUF_POS.
No Value.

OM1M_MEM_ALLOC
This callback is made when the driver is out of memory to allocate and is attempting to use the application's heap. The application must return a segment value to OM1M_MEM_ALLOC.
If the application is uses Borland allocations, the programmer must be aware that the Borland heap manager allocates 4 extra bytes for its internal management and the blocks it gives are in the form SEGMENT:0004, so the programmer must allocate 12 bytes more than requested and add one to the segment returned by Borland. To free the block, subtract one from the segment value and put 4 in the offset. For other compilers, a similar mechanism probably must be used.
For example:

```
{
    void far *Ptr=farmalloc(Value+12); // allocate 12 extra bytes
    if (!Ptr) return 0;
    return FP_SEG(Ptr)+1; // return segment+1
}
```

Value (DWORD) is the size of memory block to allocate. Return value is the segment value or NULL if the allocation fails.

OM1M_MEM_FREE
The application can now release or reuse the memory. The driver sends this message when it is done using the application heap.
For example:

```
{
    farfree(MK_FP(Value-1,4)); // free the block (Segment-1):4
    return 0;
}
```

Value (WORD) is the segment to free.
Return value is 0 if successful.

OM1M_PRIV
This command allows the private data streams that are part of the MPEG system layer (not "user data" that are part of the video layer nor "ancillary data" that are part of the audio layer) to be passed back to the application. Private data is intended only to be used for low bitrate streams which do not have strict real-time requirements.
Value is a pointer to a structure PRIVBUF that points to the buffer holding the private stream data. This buffer can be allocated by the driver; if the driver is out of memory, then it uses OM1M_MEM_ALLOC to request memory from the application's heap.Value is a pointer to a structure PRIVBUF for the private stream. It is assumed that the application allocate the

buffer, that the application consumes the data during the callback, and that upon returning the buffer may be reused.
It is assumed that the application consumes the data during the callback, and that upon returning, the buffer may be reused. It is possible for OM1M_PRIV to be sent multiple times; the private data is exhausted when size is zero.

```
typedef struct {
    WORD handle;        // starting segment of private data stream in buffer
    WORD offset;        // starting offset of private data stream in buffer
    WORD bufsize;       // amount of private data left in buffersize of the buffer
                        // if 0, signifies end-of-stream
    BYTE stream_id;     // Stream Id (1 or 2 for MPEG private data)
    DWORD PTS;          // Bits 0..31 of the Presentation Time Stamp
    DWORD DTS;          // Bits 0..31 of the Decoding Time Stamp
    BYTE Flags;         // bit 0 flags whether PTS is meaningful
                        // bit 1 is bit 32 of PTS
                        // bit 2 flags whether DTS is meaningful
                        // bit 3 is bit 32 of DTS
    } PRIVBUF;
```

OM1_CAPTURE

If OM1I_DRV_CAPS indicates that the driver supports OM1_CAPTURE, then this funtion allows the application to capture the currently displayed picture into a buffer. The capture format is RGB888 with no header. This command only works when the stream is paused. Since a 352x240 image requires a 247.5 KB buffer, the OM1I_VID_CAP_POS & OM1I_VID_CAP_SIZE settings allow the image to be captured piecemeal using smaller buffers.

Parameters

BH ⇨ OM1_CAPTURE
BL ⇨ handle of the stream
DX:AX ⇨ Pointer to the buffer where the driver should store the bitmap.
If this pointer is 0, the driver returns the size necessary to store the image. The application can then allocate a buffer of this size, set OM1I_VID_CAP_POS and OM1I_VID_CAP_SIZE, and call OM1_CAPTURE with the pointer.

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_DOS
A DOS error occurred while closing the stream. The DOS error code can be read in BL.
OM1E_INVALID_CMD
Unrecognized command code.

OM1_CLOSE

OM1_CLOSE closes a previously opened stream. All buffers are released, the file is closed, and the handle becomes invalid until associated with another stream. If the stream is not in the stop mode, a stop command is issued before closing.

Parameters

BH ⇨ OM1_CLOSE
BL ⇨ handle of the stream to close

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.

OM1E_DOS
A DOS error occurred while closing the stream. The DOS error code can be read in BL.

OM1E_INVALID_CMD
Unrecognized command code.

OM1_COPY_TO_OVERLAY

If OM1I_DRV_CAPS indicates that the application must use OM1_COPY_TO_OVERLAY, then this function must be used to update any portion of the frame buffer, including the portion which is overlaying the MPEG video data. If not, then support of this function is optional - that is, the function may or may not be implemented, and if it is, then the application may or may not choose to use it.

The bitmap can be of any size. Each line of the bitmap should be padded to end at a 4-byte boundary. Any pixels that match the current color key value are made transparent. The bitmap should be in a format compatible with the current VGA mode:

VGA mode	**bitmap format**
256 colors	1 byte per pixel, using current VGA palette
32k colors	2 bytes per pixel, RGB555
64k colors	2 bytes per pixel, RGB565

Parameters

BH ⇨ OM1_COPY_TO_OVERLAY
CX ⇨ 0 to check if the function is supported
1 to copy the bitmap to the screen
DX,AX ⇨ pointer to OM1_COPY_STRUCT (ignored if function is not supported or if CX is 0)

```
typedef struct {
        SHORT   xPosition;      // Position of dest rectangle relative
        SHORT   yPosition;      // to screen (upper left corner is 0,0)
        SHORT   width;                  // Size of destination rectangle
        SHORT   height;
        VOID            *lpData;                        // Far pointer to bitmap data
} _OM1_COPY_STRUCT;
```

Return values

BH ⇨ Zero if successful, or if function is supported.

Error codes (BH)

OM1E_INVALID_CMD
Unrecognized command code.

OM1_FREEZE

OM1_FREEZE freezes the last picture displayed of a stream in play mode. Audio continues to play. The stream counter continues to increment. The stream is in this mode until the position specified in the last OM1_PLAY command issued to this stream is reached.

Parameters

BH ⇨ OM1_FREEZE
BL ⇨ handle of the stream

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.
OM1E_INVALID_CMD
Unrecognized command code.

OM1_GET

OM1_GET gets a parameter of a stream, or the driver if the application specifies a null handle. The driver settings include information, status and default settings.

Please refer to '**Settings**' for more details.

Parameters

BH ⇨ OM1_GET
BL ⇨ handle of the stream or zero for the driver settings
CX ⇨ index of the value to get (refer to Stream settings)

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code
DX,AX ⇨ value of the setting

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.

OM1E_INDEX
The index is invalid.

OM1E_TYPE
The index represents a value meaningless for the stream (for example, a volume setting for a video stream).

OM1E_INVALID_CMD
Unrecognized command code.

OM1_GROUP

OM1_GROUP allows the application to select specific audio or video streams within a systems-multiplexed MPEG stream. Here the word "group" is equivalent to an MPEG system stream. Audio and video streams are sub-streams of the MPEG system stream.

The application may select audio streams with IDs from 0 to 31, or video streams with IDs from 0 to 15. If the ID is all ones (0xFFFF), then all streams are selected or unselected. Most hardware today can play back only one audio stream and one video stream at any time; therefore the last stream that is selected is the one that is actually played. Behavior is undetermined when all streams are simultaneously selected.

By default, audio stream 0 and video stream 0 are selected when a system stream is opened.

Parameters

BH ⇨ OM1_GROUP
BL ⇨ handle of the stream
CX ⇨ sub-command. One of the following :
- OM1F_SELECT_AUD ⇨ selects an audio stream in the group for playing
- OM1F_SELECT_VID ⇨ selects a video stream in the group for playing
- OM1F_UNSELECT_AUD ⇨ unselects an audio stream in the group from playing
- OM1F_UNSELECT_VID ⇨ unselects a video stream in the group from playing

AX ⇨ ID of the element. If all ones, the select and unselect commands apply to all streams in the group. If all streams are selected, then the ones which are actually presented depends upon implementation. Legal values are 0 to 31 plus 0xFFFF (all 1s) for audio and 0 to 15 plus 0xFFFF (all 1s) for video

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.

OM1E_ID
The stream ID is invalid.

OM1E_INVALID_CMD
Unrecognized command code.

OM1_INIT

OM1_INIT re-initializes the driver by closing any opened streams, resetting the hardware, and resetting default values. This command is useful for resetting any values which might have been changed by another application.

Parameters

BH ⇨ OM1_INIT

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_INVALID_CMD
Unrecognized command code.
OM1E_NOT_INIT
Driver not initialized.

OM1_OPEN

OM1_OPEN opens and prepares a new stream. The handle returned identifies the stream and is needed for all the commands that manipulate that stream.

If the application is using buffered streams, its callback function may receive numerous messages from OM1_OPEN in order to determine what type of stream is being opened. The first message in this case is OM1M_BUF_CREATE. This allows the application to allocate buffers or provide pointers to existing buffers.

If the operation is successful, the stream is initialized, seeked to start, and put in the stop mode. The stream settings are initialized to the default settings of the driver.

Note that it is legal to open the same file multiple times.

Parameters

BH ⇨ OM1_OPEN

CX ⇨ Type of the stream. One of the following values :

- ❒ OM1F_FILE ⇨ stream read from file
- or OM1F_BUFFERS ⇨ stream provided by application. Cannot be used with OM1F_NOACCESS

Format of the filename string when opening a file

- ❒ OM1F_PASCAL ⇨ the filename string uses the Pascal-string convention (default is C-string convention).
- ❒ OM1F_NOACCESS ⇨ the file will not be prefilled and identified now but when the stream is played. Useful for CD-ROM play. Cannot be used with OM1F_BUFFERS.

DX:AX ⇨ pointer to the filename

Return values

BH ⇨ error code or zero if successful

BL ⇨ error sub-code

AL ⇨ handle of the stream or zero if an error occurrs

Error codes (BH)

OM1E_DOS
: A DOS error occurred while opening and reading the stream. The DOS error code can be read in BL.

OM1E_TOO_MANY
: Too many streams are open and the driver cannot open another one.

OM1E_OUT_OF_MEM
: The driver can't allocate buffers for the stream.

OM1E_INVALID_FLAGS
: Flags are invalid or incoherent.

OM1E_NO_CALLBACK
: No callback function has been installed.

OM1E_INVALID_CMD
: Unrecognized command code.

OM1_PAUSE

OM1_PAUSE pauses a stream. Audio is stopped and muted, while video is frozen to the last picture and the display window is kept open. The stream counter is stopped.

Parameters

BH ⇨ OM1_PAUSE
BL ⇨ handle of the stream

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.
OM1E_INVALID_CMD
Unrecognized command code.

OM1_PLAY

OM1_PLAY plays a stream from its current position to another position given in the stream time format. The stream selected for playing has the priority on the hardware resources. If there are other streams playing, they enter the pause or stop mode (determined by the OM1F_END_PAUSE or OM1F_END_STOP flag; if OM1F_END_REPEAT is set, then the stream enters the pause mode) to let the one selected play, unless the application specifies the OM1F_WAIT flag. In this case, the new stream will wait until the present stream finishes playing. OM1F_WAIT can be used to link sequences.

OM1_PLAY should return immediately.

Parameters

BH ⇨ OM1_PLAY
BL ⇨ handle of the stream to play
CX ⇨ combination of the flags:

- ❒ OM1F_POS_END ⇨ play to the end
  or OM1F_POS_SET ⇨ play to an absolute position
  or OM1F_POS_CUR ⇨ play to a relative position from the current one
- ❒ OM1F_END_PAUSE ⇨ after playing, the stream enters pause mode on the last picture
  or OM1F_END_STOP ⇨ after playing, the stream enters stop mode
  or OM1F_END_KEEP ⇨ after playing, the stream returns in the mode it was before being played
  or OM1F_END_REPEAT ⇨ after playing, the stream restarts from where it began playing
- ❒ OM1F_WAIT ⇨ new stream waits until the present stream finishes playing.

DX,AX ⇨ position to play to in the current stream time format (if required by OM1F_POS_SET or OM1F_POS_CUR). This should be at a point after the current position.

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.

OM1E_POS
The position given is invalid.

OM1E_DOS

A DOS error occurred while playing the stream. The DOS error code can be read in BL.

OM1E_STREAM

The stream contains invalid data.

OM1E_INVALID_FLAGS

Flags are invalid or incoherent

OM1E_INVALID_CMD

Unrecognized command code.

OM1_SEEK

OM1_SEEK seeks to a position in a stream. The position is given in the stream time format. Audio for the stream under seek is muted during seek. If the application calls OM1_SEEK on stream X and stream X is displayed in the pause mode, then the same picture is displayed during and after seek.

Parameters

BH ⇨ OM1_SEEK
BL ⇨ handle of the stream to seek
CX ⇨ combination of the flags:
- ❒ OM1F_POS_START ⇨ seek to the start
- or OM1F_POS_SET ⇨ seek to an absolute position
- or OM1F_POS_END ⇨ seek to the end
- or OM1F_POS_CUR ⇨ relative seek from the current position
- ❒ OM1F_END_PAUSE ⇨ after seeking, the stream enters pause mode on the new picture
- or OM1F_END_STOP ⇨ after seeking, the stream enters stop mode
- or OM1F_END_KEEP ⇨ after seeking, the stream returns in the mode it was before. If it was in pause mode, the display is not modified and the stream returns to pause mode. If it was in stop mode, it returns to stop mode. If it was in play mode, it returns to play mode.

DX,AX ⇨ position to seek at in the current stream time format (if required by OM1F_POS_SET or OM1F_POS_CUR)

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.

OM1E_POS
The position given is invalid.

OM1E_DOS
a DOS error occurred while reading the stream. The DOS error code can be read in BL.

OM1E_STREAM
The stream contains invalid data .

OM1E_INVALID_FLAGS
Flags are invalid or incoherent.

OM1E_INVALID_CMD
Unrecognized command code.

OM1_SET

OM1_SET sets a parameter of a stream, or the driver if the application specifies a null handle. Driver settings include information, status and default settings.

The application can specify audio or video settings to the driver, with a zero handle. In this case, they become the default values - i.e. these apply to any subsequently opened streams. If the application specifies the flag OM1F_UPDATE_ALL when setting the driver, all currently opened streams take the new setting.

When setting a stream, if the application doesn't specify a flag (zero in CX), the value is updated. If the application wants to change several values and update in one shot, it can specify the flag OM1F_DONT_UPDATE. When it is ready, it specifies the flag OM1F_UPDATE_ALL with an index of zero. During the update, error messages are returned after each "set" command.
DX:AX will contain the result code corresponding to the parameter currently being set. If the application happens to change settings on a deferred basis but never calls OM1F_UPDATE_ALL, then the behavior of the driver is unpredictable.

Please refer to **'Settings'** for more details.

Parameters

BH ⇨ OM1_SET
BL ⇨ handle of the stream or zero for the driver settings
CX ⇨ index of the value to set or zero for nothing; can be combined with a flag:
- ⇨ for a stream (valid handle)
  - OM1F_DONT_UPDATE ⇨ the stream setting update is deferred
  - OM1F_UPDATE_ALL ⇨ all the stream settings are updated
- ⇨ for a driver setting (BL is 0)
  - OM1F_UPDATE_ALL ⇨ the value is passed to all opened streams

DX,AX ⇨ value

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code
DX,AX ⇨ previous value of the setting

Error codes (BH)

OM1E_HANDLE
: The handle of the stream is not valid.

OM1E_INDEX
: The index is invalid.

OM1E_ITEM_INDEX
: The index represents a value meaningless for this stream (for example a volume setting for a video stream).

OM1E_VALUE
: The value is invalid.

OM1E_WRITE
: The value cannot be written but only read.

OM1E_INVALID_FLAGS
: Flags are invalid or incoherent.

OM1E_INVALID_CMD
: Unrecognized command code.

OM1_SIGNAL

The OM1_SIGNAL command lets you specify a signal when the stream reaches a position or periodic signals given in the stream time format. Signals will be sent to the callback function of the stream or to the default callback function.

Parameters

BH ⇨ OM1_SIGNAL
BL ⇨ handle of the stream
CX ⇨ one of the flags
- OM1F_SIG_REMOVE ⇨ removes a signal (signal number given in AX)
- OM1F_SIG_REMOVE_AT ⇨ removes all the signals at the given position in AX
- OM1F_SIG_REMOVE_ALL ⇨ removes all the signals
- OM1F_SIG_AT ⇨ a signal will occur at the given position in AX
- OM1F_SIG_EVERY ⇨ signals will occur at the period in AX

DX,AX ⇨ position in the stream time format (for OM1F_SIG_AT and OM1F_SIG_REMOVE_AT)
period in the stream time format (for OM1F_SIG_EVERY)
signal number (for OM1F_SIG_REMOVE)

Return values

BH ⇨ Error code or zero if successful.
BL ⇨ Error sub-code.
AX,DX ⇨ Signal number returned when OM1_SIG_AT or OM1_SIG_EVERY specified.

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.

OM1E_VALUE
The value is invalid.

OM1E_INVALID_FLAGS
Flags are invalid or incoherent.

OM1E_TIME_FMT
Time format incorrect.

OM1E_TOO_MANY_SIGS
Too many signals are set.

OM1E_NO_CALLBACK
No callback function has been installed.

OM1E_INVALID_CMD
Unrecognized command code.

OM1_STEP

OM1_STEP advances a video stream one or more I, P, or B pictures forward. Audio is muted. To continuously step, the application must issue OM1_STEP multiple times; it is up to the application to provide any time delay before issuing another command. The stream enters the pause mode after the step.

Parameters

BH ⇨ OM1_STEP
BL ⇨ handle of the stream
CX ⇨ flags:
- ❒ OM1F_END_PAUSE ⇨ after the step, the stream enters pause mode on the picture.
- or OM1F_END_STOP ⇨ after the step, the stream stops.
- or OM1F_END_KEEP ⇨ after the step, the stream returns in the mode it was before.

DX,AX ⇨ Number of pictures to step.

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.

OM1E_DOS
A DOS error occurred while reading the stream. The DOS error code can be read in BL.

OM1E_STREAM
The stream contains invalid data.

OM1E_INVALID_FLAGS
Flags are invalid or incoherent

OM1E_INVALID_CMD
Unrecognized command code.

OM1_STOP

OM1_STOP stops a stream and closes its window if it is a video stream. The stream pointer is stopped. The stream enters stop mode.

Parameters

BH ⇨ OM1_STOP
BL ⇨ handle of the stream to close

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_HANDLE
The handle of the stream is not valid.
OM1E_INVALID_CMD
Unrecognized command code.

OM1_UNLOAD

OM1_UNLOAD removes the driver from memory. The application can use this command only if the driver was loaded in memory by the application. The application must not use any other command of the driver after sending OM1_UNLOAD.

Parameters

BH ⇨ OM1_UNLOAD

Return values

BH ⇨ error code or zero if successful
BL ⇨ error sub-code

Error codes (BH)

OM1E_DOS
A DOS error occurred while removing the driver from memory.
OM1E_INVALID_CMD
Unrecognized command code.

OM1_UPDATE

OM1_UPDATE is provided for compatibility between the playback board and some PC hardware. The command is not stream dependent and instead of a stream handle, the application puts a sub-function number in BL. The defined sub-functions are :

OM1I_UPD_PALETTE

Defined to fix the problem with some bad VGA local bus boards which don't reflect palette changes on the ISA bus. Applications mustshould use this or a similar VESA BIOS extension command every time they change the VGA palette to insure a correct change with those VGA boards. Note that OM1I_UPD_PALETTE may be used to perform palette animation, and should be coded as tightly as possible. Applications developers should recognize that there may be performance issues related to using this command.

Parameters

BH ⇨ OM1_UPDATE
BL ⇨ OM1I_UPD_PALETTE
AX ⇨ first palette index to change (default 0)
DX ⇨ number of colors to change (default 0 for 256 colors to update)

No return value, no error code for sub-function.

OM1I_UPD_VGA_MODE

Applications must use this or a similar VESA BIOS extension command every time they change resolution.

Parameters

BH ⇨ OM1_UPDATE
BL ⇨ OM1I_UPD_VGA_MODE
AX ⇨ X resolution of the mode
DX ⇨ Y resolution of the mode
CX ⇨ number of bits per pixel (typically : 4,8,15,16 or 24)
No return value, no error code for sub-function.

Error codes (BH)

OM1E_INVALID_CMD
Unrecognized command code for OM1_UPDATE.

Settings

The following are the different settings and status for streams. They can be read with the OM1_GET command and written with the OM1_SET command using the OM1I_xxx index.

The driver settings are information or default settings that will be taken by further opened streams. Driver settings can also be used to update in one shot all the opened streams (see OM1_SET and the flag OM1F_UPDATE_ALL).

Some settings are read only and are marked as 'r', others can be written and are marked 'r/w'‘. ‘R/s' means that the application can write the value only if it is not yet determined, i.e. it can set the value only once. All the writeable settings can be used as driver settings.

If the application specifies audio or video settings for a group stream, the values will be passed to all the corresponding audio and video streams of the group.

The OM1_GET and OM1_SET commands always use 32 bit values. When a value is less than 32 bits long, the more significant bits are zero.

Driver specific settings

OM1I_DRV_PRODUCT	r	pointer to the driver name
OM1I_DRV_VERSION	r	version number ❒ AX : major ❒ DX : minor
OM1I_DRV_MAX_CHAN	r	number of video and audio channels (streams) that can be played simultaneously. ❒ AX : maximum number of video channels ❒ DX : maximum number of audio channels
OM1I_DRV_HRD_STAT	r	State of the hardware. Zero for OK or a combination of these flags: ❒ OM1F_HRD_NO_DMA ⇨ no DMA channel available or OM1F_HRD_NO_INT ⇨ no interrupt available or OM1F_HRD_NO_PORT ⇨ no port available or OM1F_HRD_NOT_FOUND ⇨ board not found or OM1F_HRD_UNKNOWN ⇨ hardware problem (not yet specified)
OM1I_DRV_AUD_SUP	r	32 bit mask containing all the audio formats supported (up to 32). See OM1I_AUD_TYPE for the different formats
OM1I_DRV_MEMOR	r	Memory left in the driver memory pool, in bytes.

Y		
OM1I_DRV_CAPS	r	Driver capabilities. Zero for none or a combination of these flags: ❒ OM1F_CAPS_KEY_VID_MINMAX ⇨ key on video minimum and maximum values supported or OM1F_CAPS_KEY_ MASK ⇨ key mask supported or OM1F_CAPS_USE_COPY_TO_OVERLAY ⇨ must use OM1_COPY_TO_OVERLAY to update frame buffer or OM1F_CAPS_KEY_VID_RGB ⇨ key on video using RGB supported or OM1F_CAPS_KEY_VID_YCBCR ⇨ key on video using YCbCr supported or OM1F_CAPS_VBE20 ⇨ VBE 2.0 calls to update palette and graphics mode are hooked by the MPEG card driver or OM1F_CAN_CAPTURE ⇨ OM1_CAPTURE supported

Common settings

OM1I_STM_SOURCE	r	❒ OM1F_FILE ⇨ the stream source is a file or OM1F_BUFFERS ⇨ the stream is provided by buffers
OM1I_STM_FILEORG	r/s	Position in bytes of MPEG stream within a larger data file. For example, if all the MPEG movies are appended together, the application can tell the driver where to get the data. 32-bit value.
OM1I_STM_FILESIZE	r/s	Size in bytes of the fileMPEG stream. Use this setting when using OM1I_STM_FILEORG. 32-bit value.
OM1I_STM_MODE	r	Current mode of the stream ❒ OM1F_PAUSED ⇨ in pause mode or OM1F_STOPPED ⇨ in stop mode or OM1F_PLAYING ⇨ currently playing or OM1F_SEEKING ⇨ currently seeking or OM1F_STEPPING ⇨ currently stepping or OM1F_FROZEN ⇨ in frozen mode also defined: OM1F_READY = OM1F_PAUSED or OM1F_STOPPED
OM1I_STM_POSITION	r	The actual position of the stream in the stream time format
OM1I_STM_TIME_F	r/w	Time format of the stream

MT		❒ OM1F_BYTES: bytes format or OM1F_PICTURES: pictures format or OM1F_SAMPLES: equivalent to OM1F_PICTURES or OM1F_MSEC: milliseconds format or OM1F_HMSP: Hours (DH) Minutes (DL) Seconds (AH) Pictures (AL) or OM1F_SMPTE : SMPTE time code format (same as HMSF) or OM1F_HMSC: Hours (DH) Minutes (DL) Seconds (AH) "Cents" (1/100ths of a second) (AL) or OM1F_TIME: Same as OM1F_HMSC
OM1I_STM_USER	r/w	32 bit value which can be used to read or write any value.

Buffered streams settings

OM1I_BUF_LEFT	r	Number of bytes left in the stream buffer.
OM1I_BUF_POS	r/w	❒ DX,AX ⇨ Position causing a message, in bytes, or 0 (default) for no message.
OM1I_BUF_HANDLE	r/w	❒ AX ⇨ Segment of the buffer.
OM1I_BUF_OFFSET	r/w	The offset of the buffer address within the memory block, in bytes. Default is 0.
OM1I_BUF_SIZE	r/w	❒ DX,AX ⇨ Size of the buffer in bytes (default is zero)

Video streams settings

OM1I_VID_RATE	r	number of pictures per second ❒ AX ⇨ integer part ❒ DX ⇨ decimal part multiplied by 10000
OM1I_VID_SIZE	r	size of a picture ❒ AX ⇨ width in pixels ❒ DX ⇨ height in pixels
OM1I_VID_ASPECT	r	the pixel aspect ratio (height/width) ❒ AX ⇨ integer part ❒ DX ⇨ decimal part multiplied by 10000
OM1I_VID_BIT_RATE	r	bit rate of the bit stream in bits/second. A zero value identifies variable bit rate operation

OM1I_VID_SRC_POS	r/w	position of the source window ❒ AX ⇨ X position in pixels (default is zero) ❒ DX ⇨ Y position in pixels (default is zero)
OM1I_VID_SRC_SIZE	r/w	size of the source window ❒ AX ⇨ width in pixels (zero is default; zero sets source window width to maximum width) ❒ DX ⇨ height in pixels (zero is default; zero sets source window height to maximum height)
OM1I_VID_DEST_POS	r/w	position of the destination window ❒ AX ⇨ X position in pixels (default is zero) ❒ DX ⇨ Y position in pixels (default is zero)
OM1I_VID_DEST_SIZE	r/w	size of the destination window ❒ AX ⇨ width in pixels (zero is default; zero sets destination window width to maximum width) ❒ DX ⇨ height in pixels (zero is default; zero sets destination window height to maximum height)
OM1I_VID_KEY_MODE	r/w	The color keying mode (see "**Usage notes**" for details) ❒ OM1F_ALL_VGA : All the VGA is displayed . or OM1F_ALL_VID : All the video is displayed or OM1F_KEY_VGA : VGA keying mode (default) or OM1F_KEY_VID : Video keying mode or OM1F_KEY_MIX : combination of VGA and Video Key
OM1I_VID_KEY_COLOR_SPACE	r/w	The video keying color space mode; must be specified if either OM1F_KEY_VID or OM1F_KEY_MIX are selected, otherwise ignored. ❒ OM1F_KEY_VID_RGB ⇨ key using RGB888 or OM1F_KEY_VID_YCBCR ⇨ key using YCbCr (default)
OM1I_VID_CAP_SIZE	r/w	size of the window to be captured within the picture ❒ AX ⇨ width in pixels (zero is default; zero sets source window width to maximum width) ❒ DX ⇨ height in pixels (zero is default; zero sets source window height to maximum height)
OM1I_VID_CAP_POS	r/w	position of the window to be captured within the picture ❒ AX ⇨ X position in pixels (default is zero) ❒ DX ⇨ Y position in pixels (default is zero)
OM1I_VID_KEY_COL	r/w	Color for keying ❒ AX ⇨ index in the palette or

		❒ DL ⇨ R value ❒ AH ⇨ G value ❒ AL ⇨ B value Default is zero (black).
OM1I_VID_KEY_MIN	r/w	Minimum color value for the key on video range. 24-bit value: ❒ DL ⇨ R or Y value ❒ AH ⇨ G or Cb value ❒ AL ⇨ B or Cr value
OM1I_VID_KEY_MAX	r/w	Maximum color value for the key on video range. 24-bit value: ❒ DL ⇨ R or Y value ❒ AH ⇨ G or Cb value ❒ AL ⇨ B or Cr value
OM1I_VID_KEY_MASK	r/w	Mask for keying. ❒ AX ⇨ Mask value (default is 0x0).

Audio streams settings

OM1I_AUD_TYPE	r	type of the audio stream. ❒ OM1F_AUD_MPEG_L1 ⇨ MPEG Audio Layer I or OM1F_AUD_MPEG_L2 ⇨ MPEG Audio Layer II or OM1F_AUD_MPEG_L3 ⇨ MPEG Audio Layer III
OM1I_AUD_CHANNELS	r	❒ OM1F_AUD_STEREO ⇨ stereo or OM1F_AUD_JSTEREO ⇨ joint stereo or OM1F_AUD_DUAL ⇨ dual channel or OM1F_AUD_SINGLE ⇨ single channel
OM1I_AUD_EMPH	r	❒ OM1F_AUD_NO_EMPH ⇨ no emphasis or OM1F_AUD_EMPH_50 ⇨ 50/15 msec emphasis or OM1F_AUD_EMPH_J17 ⇨ CCITT J.17 emphasis
OM1I_AUD_RIGHTS	r	❒ OM1F_AUD_COPYRIGHT ⇨ there is a copyright on the stream or OM1F_AUD_NOCOPYRIGHT ⇨ the stream has no copyright
OM1I_AUD_ISORIGINAL	r	❒ OM1F_AUD_ORIGINAL ⇨ bitstream is an original or OM1F_AUD_COPY ⇨ bitstream is a copy
OM1I_AUD_RATE	r	the sampling rate in samples per second

OM1I_AUD_BIT_RATE	r	bit rate of the stream in bits/second
OM1I_AUD_VOLUME	r/w	volumes of the right and left channels in a linear scale ❒ AX ⇨ left channel in percent . Maximum = 100% (default) ❒ DX ⇨ right channel in percent. Maximum = 100% (default)
OM1I_AUD_BAL_L	r/w	left output channel balance: ❒ AX ⇨ percentage of left input channel. 100% (default) ❒ DX ⇨ percentage of right input channel. 0% default
OM1I_AUD_BAL_R	r/w	right output channel balance: ❒ AX ⇨ percentage of left input channel. 0% (default) ❒ DX ⇨ percentage of right input channel. 100% default

Note: All audio settings are assumed to be linear (as opposed to log or other types of scale).

Symbols

```
#ifndef __OM1MACS_H
#define __OM1MACS_H

// some macros for an easier writing of calls to the driver

#define OM1Open(Flags,Filename)                                 OM1Command(OM1_OPEN,0,Flags,Filename)

#define OM1Close(hStream)                                       OM1Command(OM1_CLOSE,hStream,0,0)
#define OM1Set(hStream, Index, Value)                           OM1Command(OM1_SET,hStream,Index,Value)
#define OM1Get(hStream,Index)                                   OM1Command(OM1_GET,hStream,Index,0)
#define OM1Play(hStream,Flags,Position)                         OM1Command(OM1_PLAY,hStream,Flags,Position)
#define OM1Seek(hStream,Flags,Position)                         OM1Command(OM1_SEEK,hStream,Flags,Position)
#define OM1Pause(hStream)                                       OM1Command(OM1_PAUSE,hStream,0,0)
#define OM1Stop(hStream)                                        OM1Command(OM1_STOP,hStream,0,0)
#define OM1Group(hStream,Flags,Value)                           OM1Command(OM1_GROUP,hStream,Flags,Value)
#define OM1Callback(hStream,Value)                              OM1Command(OM1_CALLBACK,hStream,OM1F_C,Value)
#define OM1Unload()                                             OM1Command(OM1_UNLOAD,0,0,0)
#define OM1Init()                                               OM1Command(OM1_INIT,0,0,0)
#define OM1CopyToOverlay()                                      OM1Command(OM1_COPY_TO_OVERLAY,0,Flags,OM1_COPY_STRUCT
#define OM1Freeze                                               OM1Command(OM1_FREEZE,hStream,0,0)

#define QUAD(a,b,c,d)                                           MAKEDWORD(MAKEWORD(d,c),MAKEWORD(b,a))

#endif

#ifndef __OM1FCTS_H

#define __OM1FCTS_H

#ifndef __TYPES_H

#include "types.h"
#endif

#ifdef __cplusplus

extern "C"
{
#endif
BYTE            FindDriver(void);
DWORD           OM1Command(BYTE Command,BYTE hStream,WORD Flags,DWORD Value);
#ifdef __cplusplus
}
#endif

extern WORD OM1Status;

#endif

// Commands

#define OM1_OPEN                                0x01

#define OM1_CLOSE                               0x02
#define OM1_PLAY                                0x03
#define OM1_PAUSE                               0x04
#define OM1_STOP                                0x05
#define OM1_SEEK                                0x06
#define OM1_STEP                                0x07
#define OM1_GROUP                               0x08
#define OM1_SET                                 0x09
#define OM1_GET                                 0x0A
#define OM1_CALLBACK                            0x0B
#define OM1_SIGNAL                              0x0C
#define OM1_UNLOAD                              0x0D
#define OM1_INIT                                0x0E
```

```
#define OM1_CAPTURE                     0x0F
#define OM1_UPDATE                      0x10
#define OM1_COPY_TO_OVERLAY             0x11
#define OM1_FREEZE                      Ox12
#define OM1_OEM                         0x13
#define OM1_OEMRESERVED                 0x14
#define OM1_RESERVED                    0x15
#define OM1_RESERVED                    0x16
#define OM1_RESERVED                    0x17
#define OM1_RESERVED                    0x18
#define OM1_RESERVED                    0x19

//Errors

#define OM1E_DOS                        0x0100

#define OM1E_INVALID_FLAGS              0x0200
#define OM1E_HANDLE                     0x0300
#define OM1E_NOT_IMPLEMENT              0x0400
#define OM1E_INVALID_CMD                0x0500
#define OM1E_OUT_OF_MEM                 0x0600
#define OM1E_INDEX                      0x0700
#define OM1E_TYPE                       0x0800
#define OM1E_WRITE                      0x0900
#define OM1E_TOO_MANY                   0x0A00
#define OM1E_ITEM_INDEX                 0x0B00
#define OM1E_ITEM_HANDLE                0x0C00
#define OM1E_ERROR                      0x0D00 // used for all other errors
#define OM1E_STREAM                     0x0E00
#define OM1E_NOT_CDXA_DRV               0x0F00
#define OM1E_HARDWARE                   0x1000
#define OM1E_NA                         0x1100
#define OM1E_VALUE                      0x1200
#define OM1E_TIME_FMT                   0x1300
#define OM1E_ID                         0x1400
#define OM1E_POS                        0x1500
#define OM1E_TOO_MANY_SIGS              0x1600
#define OM1E_NO_CALLBACK                0x1700
#define OM1E_NOT_INIT                   0x1800
#define OM1E_RESERVED                   0x1900
#define OM1E_RESERVED                   0x1A00
#define OM1E_RESERVED                   0x1B00
#define OM1E_RESERVED                   0x1C00
#define OM1E_OEMRESERVED                0x1D00
#define OM1E_OEMRESERVED                0x1E00
#define OM1E_OEMRESERVED                0x1F00

//Messages

#define OM1M_BUF_POS                    0x01

#define OM1M_BUF_EMPTY                  0x02
#define OM1M_BUF_SEEK                   0x03
#define OM1M_BUF_CREATE                 0x04
#define OM1M_BUF_CLOSE                  0x05
#define OM1M_BUF_TOTALSIZE              0x06
#define OM1M_COMPLETED                  0x07
#define OM1M_CANCELED                   0x08
#define OM1M_ERROR                      0x09
#define OM1M_MEM_ALLOC                  0x0A
#define OM1M_MEM_FREE                   0x0B
#define OM1M_PRIV                       0x0C
#define OM1M_RESERVED                   0x0D
#define OM1M_RESERVED                   0x0E
#define OM1M_OEMRESERVED                0x0F


// Flags

#define OM1F_PASCAL                     0x1000

#define OM1F_C                          0x2000
```

```
#define OM1F_FILE                       0x0001
#define OM1F_BUFFERS                    0x0002
#define OM1F_NOACCESS                   0x0100

#define OM1F_POS_START                  0x0100

#define OM1F_POS_SET                    0x0200
#define OM1F_POS_END                    0x0300
#define OM1F_POS_CUR                    0x0400

#define OM1F_DONT_UPDATE                0x1000

#define OM1F_UPDATE_ALL                 0x2000

#define OM1F_SIG_AT                     0x0001

#define OM1F_SIG_EVERY                  0x0002
#define OM1F_SIG_REMOVE                 0x0003
#define OM1F_SIG_REMOVE_AT              0x0004
#define OM1F_SIG_REMOVE_ALL             0x0005

#define OM1F_HRD_NO_DMA                 0x0001

#define OM1F_HRD_NO_INT                 0x0002
#define OM1F_HRD_NO_PORT                0x0004
#define OM1F_HRD_NOT_FOUND              0x0008
#define OM1F_HRD_UNKNOWN                0x0010
#define OM1F_HRD_RESERVED               0x0012
#define OM1F_HRD_RESERVED               0x0014
#define OM1F_HRD_RESERVED               0x0018
#define OM1F_HRD_OEM                    0x0020
#define OM1F_HRD_OEM                    0x0022
#define OM1F_HRD_OEM                    0x0024
#define OM1F_HRD_OEM                    0x0028

#define OM1F_END_PAUSE                  0x0000
#define OM1F_END_PAUSE                  0x0001
#define OM1F_END_KEEP                   0x0002
#define OM1F_END_REPEAT                 0x0004
#define OM1F_END_STOP                   0x0008
#define OM1F_END_RESERVED               0x001x
#define OM1F_END_OEM                    0x002x
#define OM1F_WAIT                       0x1000

#define OM1F_PAUSED                     0x0001

#define OM1F_STOPPED                    0x0002
#define OM1F_PLAYING                    0x0004
#define OM1F_SEEKING                    0x0008
#define OM1F_STEPPING                   0x0010
#define OM1F_RESERVED                   0x0020
#define OM1F_FROZEN                     0x0040
#define OM1F_RESERVED                   0x0080
#define OM1F_READY                      (OM1F_PAUSED|OM1F_STOPPED)

#define OM1F_BYTES                      0x0001

#define OM1F_SAMPLES                    0x0002
#define OM1F_MSEC                       0x0003
#define OM1F_HMSP                       0x0004
#define OM1F_HMSC                       0x0005
#define OM1F_PICTURES                   OM1F_SAMPLES
#define OM1F_TIME                       OM1F_HMSC
#define OM1F_SMPTE                      OM1F_HMSP

#define OM1F_BUF_LOOP                   0x0002

#define OM1F_ALL_VGA                    0x0001
```

```
#define OM1F_ALL_VID                    0x0002
#define OM1F_KEY_VGA                    0x0004
#define OM1F_KEY_VID                    0x0008
#define OM1F_KEY_MIX                    OM1F_KEY_VGA | OM1_KEY_VID

#define OM1F_KEY_VID_RGB                0x0000
#define OM1F_KEY_VID_YCBCR              0x0001

#define OM1F_AUD_MPEG_L1                0x0002

#define OM1F_AUD_MPEG_L2                0x0003
#define OM1F_AUD_MPEG_L3                0x0004

#define OM1F_AUD_STEREO                 0x0001

#define OM1F_AUD_JSTEREO                0x0002
#define OM1F_AUD_DUAL                   0x0003
#define OM1F_AUD_SINGLE                 0x0004

#define OM1F_AUD_NO_EMPH                0x0000

#define OM1F_AUD_EMPH_50                0x0001
#define OM1F_AUD_EMPH_J17               0x0003

#define OM1F_AUD_NOCOPYRIGHT            0x0000
#define OM1F_AUD_COPYRIGHT              0x0001

#define OM1F_AUD_COPY                   0x0000
#define OM1F_AUD_ORIGINAL               0x0001

#define OM1F_SELECT_AUD                 0x0104

#define OM1F_UNSELECT_AUD               0x0105
#define OM1F_UNSELECT_ALL_AUD           0x0106

#define OM1F_SELECT_VID                 0x0204
#define OM1F_UNSELECT_VID               0x0205
#define OM1F_UNSELECT_ALL_VID           0x0206

#define OM1F_CAPS_KEY_VID_MINMAX        0x0001
#define OM1F_CAPS_KEY_MASK              0x0002
#define OM1F_CAPS_USE_COPY_TO_OVERLAY   0x0004
#define OM1F_CAPS_KEY_VID_RGB           0x0008
#define OM1F_CAPS_KEY_VID_YCBCR         0x0010
#define OM1F_CAPS_VBE20                 0x0020
#define OM1F_CAPS_CAN_CAPTURE           0x0040
#define OM1F_CAPS_RESERVED              0x0080
#define OM1F_CAPS_RESERVED              0x0100
#define OM1F_CAPS_RESERVED              0x0200
#define OM1F_CAPS_RESERVED              0x0400
#define OM1F_CAPS_RESERVED              0x0800
#define OM1F_CAPS_OEM                   0x1000
#define OM1F_CAPS_OEM                   0x2000
#define OM1F_CAPS_OEM                   0x4000
#define OM1F_CAPS_OEM                   0x8000

// Index

#define OM1I_DRV_PRODUCT                0x0101

#define OM1I_DRV_VERSION                0x0102
#define OM1I_DRV_MAX_CHAN               0x0103
#define OM1I_DRV_OEMRESERVED            0x0104
#define OM1I_DRV_AUD_SUP                0x0105
#define OM1I_DRV_OEMRESERVED            0x0106
#define OM1I_DRV_HRD_STAT               0x0107
#define OM1I_DRV_MEMORY                 0x0108
#define OM1I_DRV_CAPS                   0x0109
#define OM1I_DRV_RESERVED               0x010A
#define OM1I_DRV_RESERVED               0x010B
#define OM1I_DRV_RESERVED               0x010C
```

```
#define OM1I_DRV_RESERVED                   0x010D
#define OM1I_DRV_RESERVED                   0x010E
#define OM1I_DRV_RESERVED                   0x010F

#define OM1I_STM_SOURCE                     0x0203

#define OM1I_STM_MODE                       0x0204
#define OM1I_STM_TIME_FMT                   0x0205
#define OM1I_STM_POSITION                   0x0206
#define OM1I_STM_USER                       0x0208
#define OM1I_STM_SIZE                       0x0209
#define OM1I_STM_RESERVED                   0x020A
#define OM1I_STM_RESERVED                   0x020B
#define OM1I_STM_RESERVED                   0x020C
#define OM1I_STM_RESERVED                   0x020D
#define OM1I_STM_OEMRESERVED                0x020E
#define OM1I_STM_OEMRESERVED                0x020F
#define OM1I_STM_OEMRESERVED                0x0210
#define OM1I_STM_FILETYPE                   0x0211
#define OM1I_STM_MEMFLAGS                   0x0212
#define OM1I_STM_FILESIZE                   0x0213
#define OM1I_STM_FILEORG                    0x0214

#define OM1I_BUF_LEFT                       0x0301

#define OM1I_BUF_POS                        0x0302
#define OM1I_BUF_OFFSET                     0x0303
#define OM1I_BUF_SIZE                       0x0304
#define OM1I_BUF_MODE                       0x0305
#define OM1I_BUF_TOTALSIZE                  0x0306
#define OM1I_BUF_HANDLE                     0x0307
#define OM1I_BUF_RESERVED                   0x0308
#define OM1I_BUF_RESERVED                   0x0309

#define OM1I_VID_OEMRESERVED                0x0401
#define OM1I_VID_RATE                       0x0402

#define OM1I_VID_SIZE                       0x0403
#define OM1I_VID_ASPECT                     0x0404
#define OM1I_VID_BIT_RATE                   0x0405
#define OM1I_VID_SRC_POS                    0x0406
#define OM1I_VID_SRC_SIZE                   0x0407
#define OM1I_VID_DEST_POS                   0x0408
#define OM1I_VID_DEST_SIZE                  0x0409
#define OM1I_VID_KEY_MIN                    0x040A
#define OM1I_VID_KEY_MAX                    0x040B
#define OM1I_VID_KEY_MASK                   0x040C
#define OM1I_VID_KEY_COL                    0x040D
#define OM1I_VID_KEY_MODE                   0x040E
#define OM1I_VID_KEY_TYPE                   0x040F
#define OM1I_VID_KEY_COLOR_SPACE            0x0410
#define OM1I_VID_CAP_POS                    0x0411
#define OM1I_VID_CAP_SIZE                   0x0412
#define OM1I_VID_RESERVED                   0x0413
#define OM1I_VID_RESERVED                   0x0414
#define OM1I_VID_RESERVED                   0x0415
#define OM1I_VID_RESERVED                   0x0416
#define OM1I_VID_RESERVED                   0x0417
#define OM1I_VID_OEMRESERVED                0x0418
#define OM1I_VID_OEMRESERVED                0x0419

#define OM1I_AUD_TYPE                       0x0501

#define OM1I_AUD_RATE                       0x0502
#define OM1I_AUD_VOLUME                     0x0503
#define OM1I_AUD_BIT_RATE                   0x0504
#define OM1I_AUD_CHANNELS                   0x0507
#define OM1I_AUD_EMPH                       0x0508
#define OM1I_AUD_RIGHTS                     0x0509
#define OM1I_AUD_SHIFT                      0x0510
#define OM1I_AUD_BAL_L                      0x0511
#define OM1I_AUD_BAL_R                      0x0512
#define OM1I_AUD_ISORIGINAL                 0x0513
```

```
#define OM1I_AUD_RESERVED                0x0514
#define OM1I_AUD_RESERVED                0x0515
#define OM1I_AUD_RESERVED                0x0516
#define OM1I_AUD_RESERVED                0x0517
#define OM1I_AUD_OEMRESERVED             0x0518
#define OM1I_AUD_OEMRESERVED             0x0519

#define OM1I_UPD_PALETTE                 0x0001

#define OM1I_UPD_VGA_MODE                0x0007
```