Digital Media Programmer's Examples

Document Number 007-3437-002

CONTRIBUTORS

Written by Patricia Creek
Production by Kay Maitz and Ruth Christian
Engineering contributions by Brian Beach, Bent Hagemark, Angela Lai, Doug Scott, Mike Travis.
St Peter's Basilica image courtesy of ENEL SpA and InfoByte SpA. Disk Thrower image courtesy of Xavier Berenguer, Animatica.

Digital Media Programmer's Examples
Document Number 007-3437-002

Contents

List of Examples

About This Guide

Digital Media Programmer's examples contains a collection of example programs that demonstrate programming concepts in the Digital Media Libraries.

The example programs in this guide are for educational purposes only, they are not guaranteed or supported by Silicon Graphics.

Chapter 1

Digital Media Content Creation Examples

This chapter contains examples that demonstrate how to import images and audio into a movie.

Creating a Movie of Screen Snapshots

Example 1-1 demonstrates how an application can concatenate a series of screen snapshots taken from the desktop into a movie.

Example 1-1 Putting Screen Snapshots into a Movie: *snapmovie.c*

`/usr/share/src/dmedia/movie/snapmovie/snapmovie.c`

Creating a Movie Soundtrack

Example 1-2 contains a listing of *addaudio.c*, which adds a soundtrack to a movie.

Example 1-2 Adding a Soundtrack to a Movie: *addaudio.c*

`/usr/share/src/dmedia/movie/addaudio/addaudio.c`

Chapter 2

Digital Media Data Conversion Examples

This chapter contains sample applications that demonstrate how to use the digital media converters.

Color Conversion

Example 2-1 contains a listing of *lator.c,* a "color calculator" application that demonstrates how to use the Color Space Library.

Example 2-1 Color Calculator: *lator.c*

```
/*
 * lator.h
 */
void lator_red(char *);
void lator_green(char *);
void lator_blue(char *);
void lator_gofigure(char **, char **, char **,
        char **, char **, char **,
        char **, char **, char **,
        char **, char **, char **,
        char **, char **);

/*
 * lator.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <bstring.h>

#include "lator.h"

static void rgb_rgbprime(char *rgbx, char *rgbx_prime);
static void rgb_ycrcb(char *rgbx, char *ycrcb);
```

```
/*
static void ycc601toyccjfif(char *ycc601, char *yccjfif);
*/
void matmul(float a[3], float m[3][3], float b[3]);
static void rgb_xyz(char *rgbx, float xyz[3]);
static void xyz_xy(float xyz[3], float xy[2]);

#define    CLAMP0(v)    (v < 0 ? 0 : (v > 255 ? 255 : v))

/* byte offset of each component in DM_IMAGE_PACKING_RGBX */
#define    RED    0
#define    GREEN    1
#define    BLUE    2
static char rgbx[4];

/* byte offset of each component in DM_IMAGE_PACKING_CbYCr */
#define    Cb    0
#define    Y    1
#define    Cr    2

/* normalized primary matrix for XYZ = NPM * RGB, SMPTE RP 176-1993 */
#define NPM_X_R                         0.4123907993
#define NPM_X_G                         0.3575843394
#define NPM_X_B                         0.1804807884

#define NPM_Y_R                         0.2126390059
#define NPM_Y_G                         0.7151686788
#define NPM_Y_B                         0.0721923154

#define NPM_Z_R                         0.0193308187
#define NPM_Z_G                         0.1191947798
#define NPM_Z_B                         0.9505321522

float npm[3][3] =
{
{ NPM_X_R, NPM_X_G, NPM_X_B },
{ NPM_Y_R, NPM_Y_G, NPM_Y_B },
{ NPM_Z_R, NPM_Z_G, NPM_Z_B },
};
```

```
static void
fill(char color[4], char *picture, int npixels)
{
    int *p = (int *)picture;

    while (npixels--) {
        *p++ = *(int *)color;
    }
}

void
lator_red(char *r)
{
    int v = atoi(r);
    rgbx[RED] = CLAMP0(v);
}

void
lator_green(char *g)
{
    int v = atoi(g);
    rgbx[GREEN] = CLAMP0(v);
}

void
lator_blue(char *b)
{
    int v = atoi(b);
    rgbx[BLUE] = CLAMP0(v);
}

static char redstr[8];
static char greenstr[8];
static char bluestr[8];
static char rpstr[8];
static char gpstr[8];
static char bpstr[8];
static char ystr[8];
static char crstr[8];
static char cbstr[8];
static char bigx_str[8];
static char bigy_str[8];
static char bigz_str[8];
static char litx_str[8];
static char lity_str[8];
```

```
void
lator_gofigure(    char **r, char **g, char **b,
        char **rp, char **gp, char **bp,
        char **y, char **cr, char **cb,
        char **bigx, char **bigy, char **bigz,
        char **litx, char **lity)
{
    char rgbxp[4], ycrcb[4];
    float xyz[3], xy[2];

    bzero(rgbxp, 4);
    bzero(ycrcb, 4);
    bzero(xyz, 3);
    rgb_rgbprime(rgbx, rgbxp);
    rgb_ycrcb(rgbxp, ycrcb);
    rgb_xyz(rgbx, xyz);
    xyz_xy(xyz, xy);

    sprintf(redstr,   "%3d", rgbx[RED]);        *r = redstr;
    sprintf(greenstr, "%3d", rgbx[GREEN]);       *g = greenstr;
    sprintf(bluestr,  "%3d", rgbx[BLUE]);        *b = bluestr;

    sprintf(rpstr, "%3d", rgbxp[RED]);        *rp = rpstr;
    sprintf(gpstr, "%3d", rgbxp[GREEN]);       *gp = gpstr;
    sprintf(bpstr, "%3d", rgbxp[BLUE]);       *bp = bpstr;

    sprintf(ystr, "%3d", ycrcb[Y]);           *y = ystr;
    sprintf(crstr, "%3d", ycrcb[Cr]);       *cr = crstr;
    sprintf(cbstr, "%3d", ycrcb[Cb]);       *cb = cbstr;

    sprintf(bigx_str, "%.3f", xyz[0]);        *bigx = bigx_str;
    sprintf(bigy_str, "%.3f", xyz[1]);        *bigy = bigy_str;
    sprintf(bigz_str, "%.3f", xyz[2]);        *bigz = bigz_str;

    sprintf(litx_str, "%.3f", xy[0]);       *litx = litx_str;
    sprintf(lity_str, "%.3f", xy[1]);       *lity = lity_str;
}

#include <dmedia/dm_params.h>
#include <dmedia/dm_image.h>
#include <dmedia/dm_color.h>
```

```
static void
rgb_rgbprime(char *rgbx, char *rgbx_prime)
{
    DMcolorconverter cvt;
    DMparams *p, *pc;

    dmColorCreate(&cvt);

    dmParamsCreate(&p);
    dmSetImageDefaults(p, 1, 1, DM_IMAGE_PACKING_RGBX);
    dmColorSetSrcParams(cvt, p);

    dmParamsCreate(&pc);
    dmParamsSetFloat(pc, DM_IMAGE_GAMMA, .45);
    dmParamsSetParams(p, DM_IMAGE_COMPONENT_ALL, pc);
    dmColorSetDstParams(cvt, p);

    dmParamsDestroy(p);
    dmParamsDestroy(pc);

    if (dmColorConvert(cvt, rgbx, rgbx_prime)) {
        printf("dmcolor error, rgb -> r'g'b'\n");
    }
    dmColorDestroy(cvt);
}

static void
rgb_ycrcb(char *rgbx, char *ycrcb)
{
    DMcolorconverter cvt;
    DMparams *p;

    dmColorCreate(&cvt);

    dmParamsCreate(&p);
    dmSetImageDefaults(p, 1, 1, DM_IMAGE_PACKING_RGBX);
    dmColorSetSrcParams(cvt, p);
    dmParamsDestroy(p);

    dmParamsCreate(&p);
    dmSetImageDefaults(p, 1, 1, DM_IMAGE_PACKING_CbYCr);
    dmColorSetDstParams(cvt, p);
    dmParamsDestroy(p);
```

```
    if (dmColorConvert(cvt, rgbx, ycrcb)) {
        printf("dmcolor error, rgb -> ycrcb\n");
    }
    dmColorDestroy(cvt);
}


static void
rgb_xyz(char *rgbx, float *xyz)
{
    float f_rgbx[3];

    f_rgbx[0] = rgbx[RED]/255.;
    f_rgbx[1] = rgbx[GREEN]/255.;
    f_rgbx[2] = rgbx[BLUE]/255.;

    matmul(xyz, npm, f_rgbx);
}

static void
xyz_xy(float xyz[3], float xy[2])
{
    float sum = xyz[0] + xyz[1] + xyz[2];

    xy[0] = xyz[0]/sum;
    xy[1] = xyz[1]/sum;
}

static void
matmul(float a[3], float m[3][3], float b[3])
{
        a[0] = m[0][0] * b[0] + m[0][1] * b[1] + m[0][2] * b[2];
        a[1] = m[1][0] * b[0] + m[1][1] * b[1] + m[1][2] * b[2];
        a[2] = m[2][0] * b[0] + m[2][1] * b[1] + m[2][2] * b[2];
}
```

Chapter 3

Digital Media File Operations Examples

This chapter contains sample applications that perform file operations using the digital media libraries.

Audio File Recognition

Example 3-1 contains a listing of *afinfo.c,* which performs audio file recognition.

Example 3-1 Audio File Recognition: *afinfo.c*

```
#include <sys/types.h>
#include <sys/stat.h>
#include <ctype.h>
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <malloc.h>
#include <audiofile.h>
#include <audioutil.h>
#include <dmedia/dm_params.h>
#include <dmedia/dm_audio.h>

void myexit(int s) {
    if(s != 0) fprintf(stderr, "\n\nexited with %d.  oserror = %d\n", s,
oserror());
    exit(s);
}

#define MAX_AWARE_OPTS 5

int legalMarkID(int* ids, int nids, int id) {
    while(nids--)
        if(*ids++ == id) return 1;
    return 0;
}
```

```
typedef struct _maxamp {
    float max[4];
    int loc[4];
    int timetag;
} maxamp;

void dump(char *, int);

main(int argc, char *argv[]) {
    AFfilehandle handle;
    int verbose = 0;
    int sampfmt, sampwidth;
    int fileformat, compression;
    int channels;
    int byteorder, numMiscs, numInsts, numMarkers, numCompTypes;
    int numSupportedInsts, numSupportedInstParams;
    int numSupportedLoops, numSupportedMarkers;
    int defaultSampleFormat, defaultSampleWidth;
    int arg = 1;
    DMparams* formatParams;
    double slope, intercept, minclip, maxclip;
    if(argc < 2) {
        fprintf(stderr, "Usage:  %s [-v] filename1 [filename2 ...]\n", argv[0]);
        myexit(0);
    }
    if(!strcmp(argv[arg], "-v")) {
        verbose = 1;
        arg++;
    }
    mallopt(M_DEBUG, 1);
    for(; arg < argc; arg++) {
        char* filename = argv[arg];
        int* mids = NULL;
        int fd = 0;
        int track = AF_DEFAULT_TRACK;
        struct stat _stat;
        _stat.st_mode = 0;    /* reset */
        if(stat(filename, &_stat) < 0 || !(_stat.st_mode & S_IFREG))
            continue;
        fprintf(stderr, "\n**** %s Format Dump ****\n", filename);
        handle = afOpenFile(filename, "r", AF_NULL_FILESETUP);
```

```
if(!handle) {
    int err;
    dmGetError(&err, NULL);
    if(err == AF_BAD_NOT_IMPLEMENTED && (fd = open(filename, 0)) > 0)
        fprintf(stderr, “Cannot read file (type %s)\n”,
            (char *) afQueryPointer(AF_QUERYTYPE_FILEFMT, AF_QUERY_DESC,
                afIdentifyNamedFD(fd, filename, NULL), 0, 0));
    if(fd > 0) close(fd);
    continue;
}
dmParamsCreate(&formatParams);
afGetFormatParams(handle, track, formatParams);
dmParamsDestroy(formatParams);
fileformat = afGetFileFormat(handle, 0);
fprintf(stderr, “File format: %s\n”,
    (char *) afQueryPointer(AF_QUERYTYPE_FILEFMT,
                            AF_QUERY_NAME, fileformat, 0, 0));
fprintf(stderr, “Duration: %f seconds\n”,
    afGetFrameCount(handle, track)/afGetRate(handle, track));
fprintf(stderr, “Framecount: %d\n”,
    afGetFrameCount(handle, track));
fprintf(stderr, “Data Offset: %d bytes\n”,
   afGetDataOffset(handle, track));
fprintf(stderr, “Total Data Size: %d bytes\n”,
   afGetTrackBytes(handle, track));
afGetSampleFormat(handle, track, &sampfmt, &sampwidth);
fprintf(stderr, “Sample Format: %s, %d bits per sample\n”,
    sampfmt == AF_SAMPFMT_TWOSCOMP ? “2’s complement”
    : sampfmt == AF_SAMPFMT_UNSIGNED ? “unsigned”
    : sampfmt == AF_SAMPFMT_FLOAT ? “floating point”
    : sampfmt == AF_SAMPFMT_DOUBLE ? “double-precision floating point”
    : “unknown”,
    sampwidth);
byteorder = afGetByteOrder(handle, track);
fprintf(stderr, “Byte order is %s-endian\n”,
       byteorder == AF_BYTEORDER_BIGENDIAN ? “big” : “little”);
fprintf(stderr, “Samprate: %f\n”, afGetRate(handle, track));
fprintf(stderr, “Channel Count:  %d\n”,
channels = afGetChannels(handle, track));
fprintf(stderr, “Frame Size: %.4f bytes\n”,
   afGetFrameSize(handle, track, 0));
afGetPCMMapping(handle, track, &slope, &intercept, &minclip, &maxclip);
fprintf(stderr,
    “PCM Mapping:\t slope: %.4f  intercept: %.4f\n\t\tmin: %.4f
                      max: %.4f\n”, slope, intercept, minclip, maxclip);
```

```
compression = afGetCompression(handle, track);
fprintf(stderr, "Compression Type: %s\n",
    (char *) afQueryPointer(AF_QUERYTYPE_COMPRESSION,
                            AF_QUERY_NAME, compression, 0, 0));
if(compression == AF_COMPRESSION_MPEG1) {
    int layer, target;
        AUpvlist pvlist = AUpvnew(MAX_AWARE_OPTS);
        AUpvsetparam  (pvlist, 0, AF_MPEG_PARAM_LAYER);
        AUpvsetvaltype(pvlist, 0, AU_PVTYPE_LONG);
        AUpvsetparam  (pvlist, 1, AF_MPEG_PARAM_BITRATE_TARGET);
        AUpvsetvaltype(pvlist, 1, AU_PVTYPE_LONG);
        afGetCompressionParams(handle, track,
                               &compression, pvlist, 2);
    AUpvgetval(pvlist, 0, &layer);
    AUpvgetval(pvlist, 1, &target);
    fprintf(stderr, "\tCompression layer %d\n",
            layer==AF_MPEG_LAYER_I ? 1 : 2);
    fprintf(stderr, "\tBit Rate: %d BPS\n", target);
    AUpvfree(pvlist);
}
else if(compression == AF_COMPRESSION_AWARE_MULTIRATE) {
    int policy, target;
    AUpvlist pvlist = AUpvnew(MAX_AWARE_OPTS);
    AUpvsetparam  (pvlist, 0, AF_AWARE_PARAM_BITRATE_POLICY);
    AUpvsetvaltype(pvlist, 0, AU_PVTYPE_LONG);
    afGetCompressionParams(handle, track,
                           &compression, pvlist, 1);
    AUpvgetval(pvlist, 0, &policy);
    fprintf(stderr, "\tCompression bitrate policy: %s\n",
       policy == AF_AWARE_FIXED_RATE ? "Fixed Rate"
       : policy == AF_AWARE_CONST_QUAL ? "Constant Quality"
       : policy == AF_AWARE_LOSSLESS ? "Lossless"
       : "Unknown!");
    if(policy == AF_AWARE_FIXED_RATE) {
            AUpvsetparam(pvlist, 0, AF_MPEG_PARAM_BITRATE_TARGET);
            AUpvsetvaltype(pvlist, 0, AU_PVTYPE_LONG);
            afGetCompressionParams(handle, track,
                                   &compression, pvlist, 1);
        AUpvgetval(pvlist, 0, &target);
        fprintf(stderr, "\tBit Rate: %d BPS\n", target);
    }
    AUpvfree(pvlist);
}
```

```
defaultSampleFormat = afQueryLong(AF_QUERYTYPE_FILEFMT,
        AF_QUERY_SAMPLE_FORMATS, AF_QUERY_DEFAULT, fileformat, 0);
defaultSampleWidth = afQueryLong(AF_QUERYTYPE_FILEFMT,
        AF_QUERY_SAMPLE_SIZES, AF_QUERY_DEFAULT, fileformat, 0);
if(verbose) {
fprintf(stderr, "Format's default sample format: %s  width: %d\n",
    defaultSampleFormat == AF_SAMPFMT_TWOSCOMP ? "2's complement"
    : defaultSampleFormat == AF_SAMPFMT_UNSIGNED ? "unsigned"
    : defaultSampleFormat == AF_SAMPFMT_FLOAT ? "floating point"
    : defaultSampleFormat == AF_SAMPFMT_DOUBLE ?
         "double-precision floating point"
    : "unknown",
    defaultSampleWidth);
fprintf(stderr, "\n");
} /* verbose */

numCompTypes = afQueryLong(AF_QUERYTYPE_FILEFMT,
        AF_QUERY_COMPRESSION_TYPES, AF_QUERY_VALUE_COUNT,
        fileformat, 0);
if(verbose) {
fprintf(stderr, "This format supports %d compression type(s)",
        numCompTypes);
if(numCompTypes > 0) {
    int i;
    int* types = (int *) afQueryPointer(AF_QUERYTYPE_FILEFMT,
                                        AF_QUERY_COMPRESSION_TYPES,
                                          AF_QUERY_VALUES, fileformat,
                                        0);
    fprintf(stderr, ":\n");
    for(i = 0; i < numCompTypes; i++)
        fprintf(stderr, "\t%s\n",
            (char *) afQueryPointer(AF_QUERYTYPE_COMPRESSION,
                            AF_QUERY_NAME, types[i], 0, 0));
    free(types);
}
fprintf(stderr, "\n");
} /* verbose */

if(verbose) {
numSupportedMarkers = afQueryLong(AF_QUERYTYPE_MARK,
        AF_QUERY_MAX_NUMBER, fileformat, 0, 0);
fprintf(stderr, "This format supports %d mark(s)",
        numSupportedMarkers);
fprintf(stderr, "\n");
} /* verbose */
```

```
numMarkers = afGetMarkIDs(handle, track, NULL);
if(numMarkers > 0) {
    int m;
    mids = (int *) calloc(numMarkers, sizeof(int));
    afGetMarkIDs(handle, track, mids);
    fprintf(stderr, "%d mark chunk(s) found:\n", numMarkers);
    for(m = 0; m < numMarkers; m++) {
        fprintf(stderr,
            "\tID: %d  name: \"%s\"\tcomment: \"%s\"\tposition: %d\n",
            mids[m], afGetMarkName(handle, track, mids[m]),
            afGetMarkComment(handle, track, mids[m]),
            afGetMarkPosition(handle, track, mids[m])
        );
    }
}
numSupportedInsts = afQueryLong(AF_QUERYTYPE_INST, AF_QUERY_MAX_NUMBER,
                            fileformat, 0, 0);
numSupportedInstParams = afQueryLong(AF_QUERYTYPE_INSTPARAM,
                AF_QUERY_ID_COUNT, fileformat,
                0, 0);
if(verbose) {
fprintf(stderr, "This format supports %d inst chunk(s)",
    numSupportedInsts);
if(numSupportedInsts > 0) {
    numSupportedLoops = afQueryLong(AF_QUERYTYPE_INST,
            AF_QUERY_LOOPS, AF_QUERY_MAX_NUMBER, fileformat, 0);
    fprintf(stderr, ", %d loop(s) per inst", numSupportedLoops);
    fprintf(stderr, ", and %d inst param(s)", numSupportedInstParams);
}
fprintf(stderr, "\n\n");
} /* verbose */

numInsts = afGetInstIDs(handle, NULL);
if(numInsts > 0) {
    int i;
    int* ids = (int *) calloc(numInsts, sizeof(int));
    afGetInstIDs(handle, ids);
    fprintf(stderr, "%d inst chunk(s) found:\n", numInsts);
    for(i = 0; i < numInsts; i++) {
        int nLoops;
        int *loops = NULL;
        int instid = ids[i];
        fprintf(stderr, "\tid %d:\n", instid);
        nLoops = afGetLoopIDs(handle, instid, NULL);
```

```
        if(nLoops > 0) {
            int l;
            fprintf(stderr, "\t%d loop(s):\n", nLoops);
            loops = (int *) calloc(nLoops, sizeof(int));
            afGetLoopIDs(handle, instid, loops);
            for(l = 0; l < nLoops; l++) {
                int mode = afGetLoopMode(handle, instid, loops[l]);
                int stID = afGetLoopStart(handle, instid, loops[l]);
                int endID = afGetLoopEnd(handle, instid, loops[l]);
                fprintf(stderr,
                  "\t\ttrackID: %d  mode: %s\tstartID: %d\tendID: %d\n",
                    afGetLoopTrack(handle, instid, loops[l]),
                    mode == AF_LOOP_MODE_NOLOOP ? "ignored" :
                    mode == AF_LOOP_MODE_FORW ? "forward" :
                        "forward/backward",
                    legalMarkID(mids, numMarkers, stID) ? stID : stID,
                    legalMarkID(mids, numMarkers, endID) ? endID : endID
                );
            }
            cfree(loops);
        }
        if(numSupportedInstParams > 0) {
            int i;
            int *paramIDs = (int *) afQueryPointer(
                AF_QUERYTYPE_INSTPARAM, AF_QUERY_IDS, fileformat, 0, 0);
            fprintf(stderr, "\t%d paramID(s):\n",
                    numSupportedInstParams);
            for(i = 0; i < numSupportedInstParams; i++)
                fprintf(stderr, "\t\tID: %d  Name: \"%s\"  Value: %d\n",
                    paramIDs[i],
                    (char *) afQueryPointer(AF_QUERYTYPE_INSTPARAM,
                                AF_QUERY_NAME,
                                fileformat,
                                paramIDs[i], 0),
                    afGetInstParamLong(handle, instid, paramIDs[i]));

            cfree(paramIDs);
        }
    }
    cfree(ids);
}
```

```
if(verbose) {
numMiscs = afQueryLong(AF_QUERYTYPE_MISC,
            AF_QUERY_TYPE_COUNT, fileformat, 0, 0);
fprintf(stderr, “This format supports %d MISC chunk type(s)”, numMiscs);
if(numMiscs > 0) {
    int i;
    int* miscs = (int *) afQueryPointer(AF_QUERYTYPE_MISC,
                                        AF_QUERY_TYPES, fileformat,
                                        0, 0);
    fprintf(stderr, “:\n”);
    for(i = 0; i < numMiscs; i++)
        fprintf(stderr, “\tType: %d  Name: \”%s\”\n”,
            miscs[i],
            (char *) afQueryPointer(AF_QUERYTYPE_MISC,
                AF_QUERY_NAME, miscs[i], 0, 0));
    free(miscs);
}
fprintf(stderr, “\n”);
} /* verbose */

numMiscs = afGetMiscIDs(handle, 0);
if(numMiscs > 0) {
    int i;
    int* miscIDs = calloc(numMiscs, sizeof(int));
    int miscType, miscSize;
    char* miscBuf = NULL;
    char* typeName = “unknown”;
    afGetMiscIDs(handle, miscIDs);
    fprintf(stderr, “Misc chunks present in file:\n”);
    for(i = 0; i < numMiscs; i++) {
        miscType = afGetMiscType(handle, miscIDs[i]);
        miscSize = afGetMiscSize(handle, miscIDs[i]);
        typeName = (char *) afQueryPointer(AF_QUERYTYPE_MISC,
                   AF_QUERY_NAME,
                                miscType, 0, 0);
        fprintf(stderr, “\tType: ‘%s’ [%d]  Size: %d\n”,
            typeName, miscType, miscSize);
        if(miscSize > 0) {
            miscBuf = calloc(miscSize, 1);
            afReadMisc(handle, miscIDs[i], miscBuf, miscSize);
        }
        if(miscType != AF_MISC_UNRECOGNIZED
                && miscType != AF_MISC_APPL
                && miscType != AF_MISC_MIDI
                && miscType != AF_MISC_PCMMAP
```

```
                        && miscType != AF_MISC_IRCAM_PEAKAMP) {
                    char c;
                    char* m = miscBuf;
                    fprintf(stderr, "\tMisc chunk text:\n\t\t\"");
                    while(miscSize-- > 0) {
                        if(*m && isascii(c = *m++)) fprintf(stderr, "%c", c);
                        else if(miscSize > 1) fprintf(stderr, " ");
                    }
                    fprintf(stderr, "\"\n");
                }
                else if (miscType == AF_MISC_IRCAM_PEAKAMP) {
                    if(miscSize >= sizeof(maxamp)) {
                        int c;
                        maxamp* amp = (maxamp *) miscBuf;
                        fprintf(stderr, "\tPeak amp information:\n");
                        for(c = 0; c < channels; c++)
                            fprintf(stderr, "\t\tChan %d: peak: %f  loc: %d\n",
                                c, amp->max[c], amp->loc[c]);
                        fprintf(stderr, "\tCreation time:  %d\n", amp->timetag);
                    }
                    else fprintf(stderr, "\t CORRUPTED MAXAMP: size %d < %d\n",
                             miscSize, sizeof(maxamp));
                }
                else {
                    fprintf(stderr, "\tMisc chunk dump:\n\n");
                    dump(miscBuf, miscSize);
                    fprintf(stderr, "\n\n");
                }
                if(miscBuf) cfree(miscBuf);
                miscBuf = NULL;
            }
            cfree(miscIDs);
        }
        cfree(mids);
        afCloseFile(handle);
    }
}
```

```
int base = 16;

void dump(char *b, int bufsize) {
    unsigned char buf[17];
    int ct, bytes, remaining = bufsize, offset = -16,cu,az = 0;

    while (remaining) {
        bytes = remaining > 16 ? 16 : remaining;
        bcopy(b, buf, bytes);
        remaining -= bytes;
        ct = bytes;
        for (cu = 0; cu != ct; cu++)
            if (buf[cu])
                break;
        if (cu == 16)
            if (az) {
                if (az == 1)
                    fprintf(stderr, "*\n");
                az = 2;
                offset += 16;
                continue;
            }
            else
                az = 1;
        else
            az = 0;
        buf[ct] = '\0';

        if (base==16)
          fprintf(stderr, "%08x: ",offset += 16);
        else if (base==8)
          fprintf(stderr, "%08o: ",offset += 16);
        else
          fprintf(stderr, "%08d: ",offset += 16);

        for (cu = 0; cu != ct; cu++)
            fprintf(stderr, ((cu & 3) == 3) ? "%02x " : "%02x",(int) buf[cu]);
        fprintf(stderr, "%*s",40-(ct*2+ct/4),"");
        for (cu = 0; cu != ct; cu++)
            if (buf[cu] < ' ' || buf[cu] >= '\x7f')
                buf[cu] = '.';
        fprintf(stderr, "%s\n",buf);
    }
}
```

Chapter 4

Digital Media I/O Examples

This chapter contains sample applications that demonstrate how to transport digital media data into and out of your workstation.

Video Capture with Compression

Example 4-1 contains a listing of *streamDecompress.c*, part of the *dmplay.dmIC* application, which demonstrates synchronized audio and video capture into a movie file. The *dmplay.dmIC* program demonstrates the use of DMbuffers and the dmIC API for video capture and compression on the O2 workstation.

Example 4-1 Capturing Compressed Video to Disk: *dmplay.dmIC*

`/usr/share/src/dmedia/dmplay/dmplay.dmic/streamDecompress.c`

Video I/O

This section contains examples that demonstrate video I/O on the O2 workstation. The directory */usr/share/src/dmedia/video/vl/DMbuffer* contains:

examples.h	include file containing definitions for this set of programs
getdevnode.h	include file containing definitions for helper functions
getdevnode.c	helper functions for this set of programs
mtov.c	memory to video I/O
vtom.c	video to memory I/O
vtos.c	video to screen I/O
vtov.c	video to video I/O

Chapter 5

Digital Media Editing and Effects Examples

This chapter contains sample applications that demonstrate how to edit media streams.

Movie Editing

This section contains *sceneDetector,* a program that detects scene transitions in an input video stream and creates a movie with a representative view of each scene. The resulting movie can then be printed to make a storyboard of an existing movie. Each scene in the storyboard is labeled with its timecode in the parent movie.

Example 5-1 Storyboarding: *sceneDetector*

`/usr/share/src/dmedia/movie/SceneDetector/Scene.c++`