

```

=====
===
===                               ===
===               Movie Library   ===
===               version 1.0     ===
===                               ===
=== (C) 1997 Sony Computer Entertainment Inc. All Rights Reserved. ===
===                               ===
=====

```

[Abstract]

Movie Library is a C++ library which compresses images into MDEC compression format. It compresses RGB and YUV into BS format. Movie Library can be used to customize movie encoding process for your own environment. Movie Library covers 5 different hardware platforms.

[Contents of this package]

```

include      include files

x86           library and binaries for Windows 95/NT
alpha         library and binaries for Windows NT(DEC alpha chip)
mac           library and binaries for Macintosh(PowerPC)
sgi           library and binaries for SGI
sparc         library and binaries for SUN sparc

sample        sample program and samle data

readme_e.txt  this file

```

[Environment]

Movie Library was built under environments listed below. Please use suitable linker which can be used under these environments.

```

x86:   Intel Pentium, Windows 95, Microsoft Visual C++ (dll)
alpha: DEC alpha, Windows NT, Microsoft Visual C++
mac:   Macintosh, PowerPC, MetroWerks CodeWarrior (CW Pro1)
sgi:   SGI IRIX 5.3, CC
sparc: SUN Solaris 2.5.1, CC

```

[Usage]

MdecBegin() is a function which initializes Movie Library. It needs to be called prior to any other library function call. MdecEnd() needs to be called to clean up Movie Library. Please refer to a sample code below.

```

MdecBegin();

for (...) {
    MdecEncode(...);
}

MdecEnd();

```

Please refer to the sample program in 'sample' directory as well to see more details.

[Structure and Functions]

```

+ structure for Q matrix
    struct DECDCTENV {
        u_char iq_y[64];    // IQ (Y): zig-zag order
        u_char iq_c[64];    // IQ (Cb,Cr): zig-zag order
        short  dct[64];     // IDCT coef (currently not available)
    };

+ input data format
    enum Format {
        RGBformat,
        YUVformat
    };

+ initialization of Movie Library
    BOOL MdecBegin();
        // Initializes library
        // return value: TRUE if succeeded

+ termination of Movie Library
    BOOL MdecEnd();
        // Terminates library
        // return value: TRUE if succeeded

+ Store Q matrix
    void MdecPutEnv(DECDCTENV *env);
        // Changes Q matrix and DCT matrix
        // DCT matrix cannot be changed with this version

+ Store default Q matrix
    void MdecPutEnvDefault();
        // Sets default Q matrix

+ Get Q matrix
    void MdecGetEnv(DECDCTENV *env);
        // Get Q matrix

+ Encode (with bs size specified)
    u_long MdecEncode(
        u_char *pix,           // image to be compressed (RGB or YUV)
        Format format,          // image format (RGBformat or YUVformat)
        u_short width,         // width of the image
        u_short height,        // height of the image
        u_long *bs,            // buffer for bs data
        u_long bsBuffSize,     // buffer size for bs data (unit: long
word)
        u_long bsLimitSize,    // maximum size for bs data (unit: byte)
        double fullness,       // fullness (0.0-1.0) ... see below
        u_short initQuant,     // initial quant value (1 - 63)
        u_short *finalQuant    // final quant value returned
    );
        // Encodes one image
        // with specifying size of encoded data(bs data).
        // return value: size of encoded data (byte)
        // This function try to encode image data into the size shown
below.
        //          fullness * bsLimitSize <= size <= bsLimitSize

+ Encode (with quantization value specified)
    u_long MdecEncode(

```

```

    u_char *pix,           // image to be compressed (RGB=8:8:8)
    Format format,         // image format
    u_short width,        // width of the image
    u_short height,       // height of the image
    u_long *bs,           // buffer for bs data
    u_long bsBuffSize,    // buffer size for bs data (unit: long
word)
    u_short quant          // initial quant value (1 - 63)
);
// Encodes one image
// with specifying quant value.
// return value: size of encoded data (byte)

```