AN1367

将 Helix MP3 解码器移植到 Microchip 32 位 PIC32MX MCU

作者： *Sunil Fernandes*
Microchip Technology Inc.

简介

MPEG-1、MPEG-2 和 MPEG-2.5 Layer 3（MP3）音频编码格式是消费音频存储设备和数字音频播放器普遍使用的音频格式。多种位速率、可变位速率和可选音频采样率等特性使此算法成为各种多媒体应用的优先选择。

本应用笔记介绍将开源 Helix MP3 解码器算法移植到 Microchip 32 位 PIC32MX 单片机（MCU）的过程。本文档提供的源代码演示了使用 Helix MP3 解码器的 MP3 播放器应用程序。该 MP3 播放器应用程序使用 Microchip 的 USB 协议栈从 USB 闪存驱动器（在本文档中称为 U 盘）读取 MP3 文件，并使用 Microchip 图形协议栈实现带触摸屏支持的图形用户界面（Graphical User Interface， GUI）。

应用程序开发人员可能需要向开源代码添加专有代码，以满足目标应用程序的要求。静态编译开源代码时，此专有代码可能受到开源最终用户许可协议的限制。在许多情况下，这对于应用程序所有者来说可能无法接受。因此，本应用笔记介绍了运行时库加载（Run-Time Library Loading， RTLL）技术，用以保护应用程序的知识产权。

本应用笔记的组织顺序如下：

1. 介绍 Helix MP3 解码器库。
2. 演示应用程序中使用的 RTLL 技术。
3. 介绍演示应用程序代码。
4. 编译和运行演示应用程序需要的步骤。

关于 HELIX MP3 解码器

Helix MP3 解码器可浮点和定点实现。将该算法移植到 PIC32MX 单片机时可考虑定点实现。该算法可运行在任意 32 位定点处理器上，并完全使用 C 语言编码，可选择用优化的汇编指令替换某些代码段。

Helix MP3 解码器提供对 MPEG-1、MPEG-2 和 MPEG-2.5 的第 3 层（Layer 3）支持。它支持可变位速率、恒定位速率，以及立体声和单声道音频格式。有关实现和特性的详细信息，请访问 Helix MP3 解码器网站：https://datatype.helixcommunity.org/Mp3dec。

Helix MP3 解码器的源代码是开源代码，受制于源代码随附文件中描述的许可协议。应该注意的是，尽管 Helix MP3 解码器可免费使用并且开源，但 MP3 算法本身并不免费，且关联有版税。必须支付这些版税才能使用此算法。有关详细信息，请访问 www.mp3licensing.com。

将 Helix MP3 解码器移植到 PIC32MX 单片机

要将 Helix MP3 解码器移植到 PIC32MX 平台，必须从 Helix MP3 解码器网站下载该解码器源代码。请按照网页上的指示下载源代码。或者，也可以使用本应用笔记提供的 Helix MP3 解码器源代码。本文档提供的 Helix MP3 解码器源代码已经过修改，可在 PIC32MX 器件上工作。要获得最新的源代码，请访问 Helix MP3 解码器网站：https://datatype.helixcommunity.org/Mp3dec。

在下载的源代码中，每个文件夹都包含三个许可文件：`RPSL.txt`、`RCSL.txt` 和 `LICENSE.txt`。我们建议用户阅读这些许可文件，并要求用户确保遵守。

Helix MP3 解码器源代码（下载自 Helix MP3 解码器网站）位于 fixpt 文件夹中。将解码器源代码移植到 PIC32MX 器件需要执行以下步骤。

1. 打开 \fixpt\real 文件夹中的 assembly.h 文件。
2. 转到针对 MIPS 平台的 FASTABS 函数定义（该文件的第 337 行）。此函数是一个 MIPS 汇编语言实现的绝对值函数。
3. 将函数体注释掉并将此部分替换为 C 语言实现（见例 1），然后保存这些更改。
4. 打开 fixpt 文件夹中的 mp3dec.c 文件，注释掉第 47 行（见例 2）。移植到 PIC32MX 的 Helix MP3 解码器不需要此功能。

这些步骤适用于 1.8 版的 assembly.h 文件和 1.6 版的 mp3dec.c 文件。

需要对下载自 Helix MP3 解码器网站的源代码进行这些修改。本文档提供的 Helix MP3 解码器源代码已包含这些修改。

例 1：

```
static __inline int FASTABS(int x)
{
//  int t=0; /* 实际上不需要初始化，只是为了避免出现警告。 */
//
//  __asm__ volatile (
//      "sra %0, %1, 31 \n\t"
//      "xor %1, %1, %0 \n\t"
//      "sub %0, %1, %0 \n\t"
//      : "=&r" (t)
//      : "r" (x)
//   );
//
//  return t;

    /* 在 MIPS M4K 内核上解码某些文件时，上面的代码会产生问题， */
    /* 因此将其注释掉。 */
    return((x > 0) ? x :-(x));
}
```

例 2：

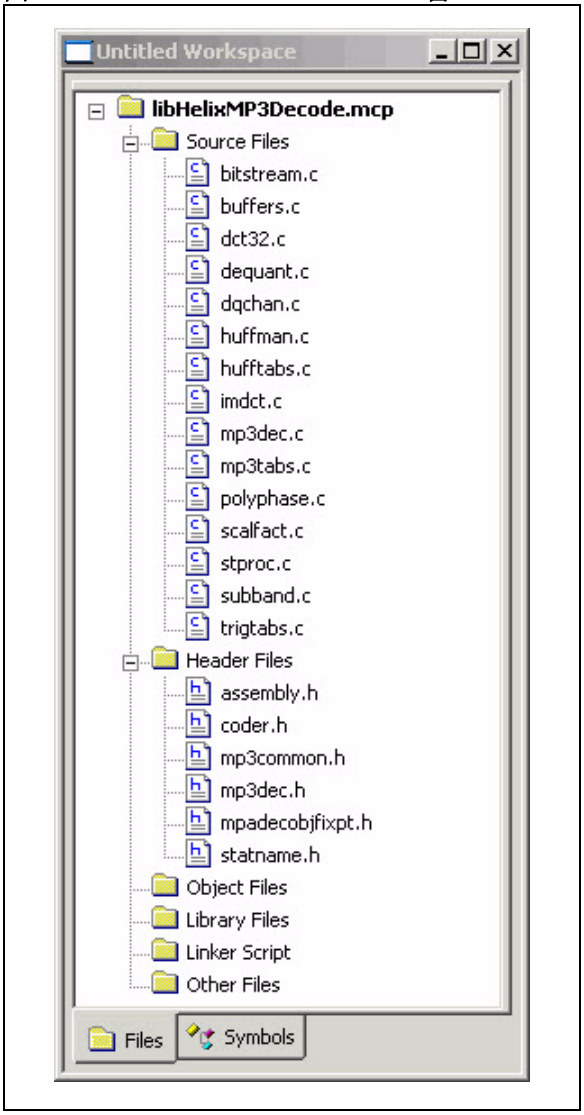
```
#include "string.h"             /* 为了使用 memmove、 memcpy（如果需要， */
                                /* 可以用不同的实现替换） */

#include "mp3common.h"          /* 包含 mp3dec.h（公用 API）和内部 */
                                /* 平台无关的 API */

//#include "hxthreadyield.h"
```

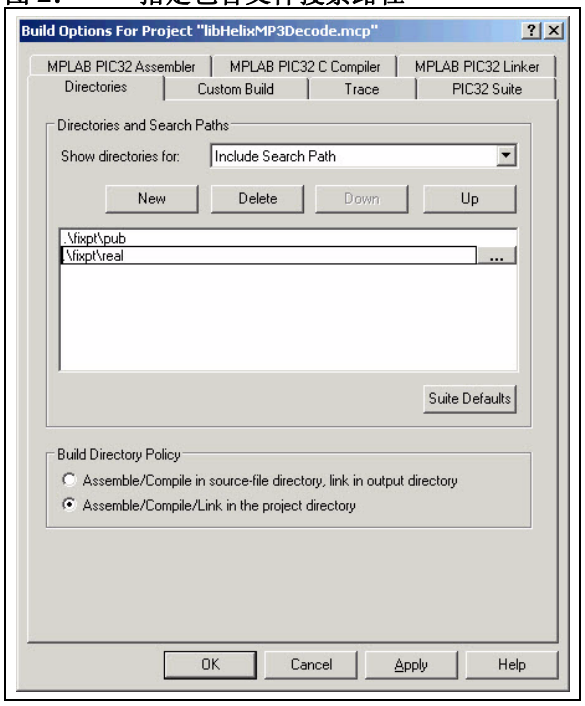
编译一个包含Helix源代码文件的项目，包括`fixpt`和`fixpt\real`文件夹中的所有`.c`源代码文件。用户可以选择是否包含项目中的所有头文件。图 1 显示了 MPLAB IDE Project Explorer（项目资源管理器）窗口中的文件列表。

图 1： PROJECT EXPLORER 窗口



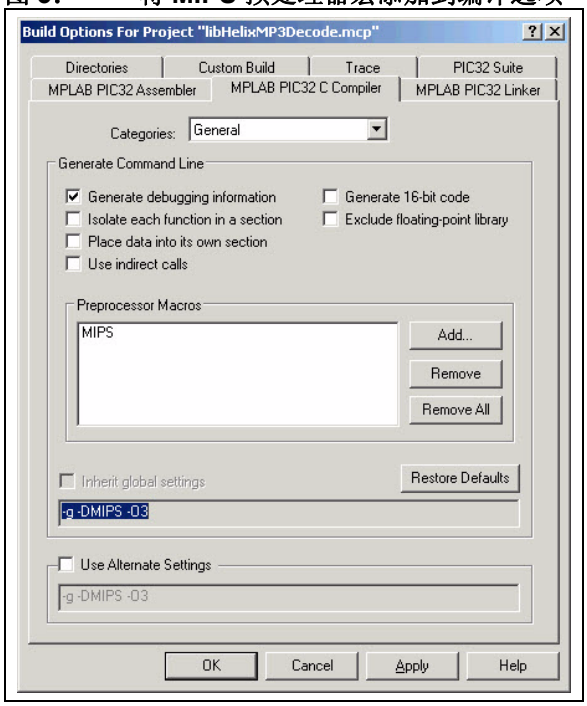
需要指定`fixpt\pub`和`fixpt\real`文件夹的路径，并且该路径应该是项目文件的相对路径。单击 ***Project>Build Options>Project***（项目 > 编译选项 > 项目）。从 **Directories**（目录）选项卡中选择 Include Search Path（包含搜索路径），并指定 MPLAB 项目位置的相对路径；或者，也可以指定绝对路径。例如，如果项目和`fixpt`文件夹在同一个文件夹中，选择的选项将与图 2 类似。

图 2： 指定包含文件搜索路径



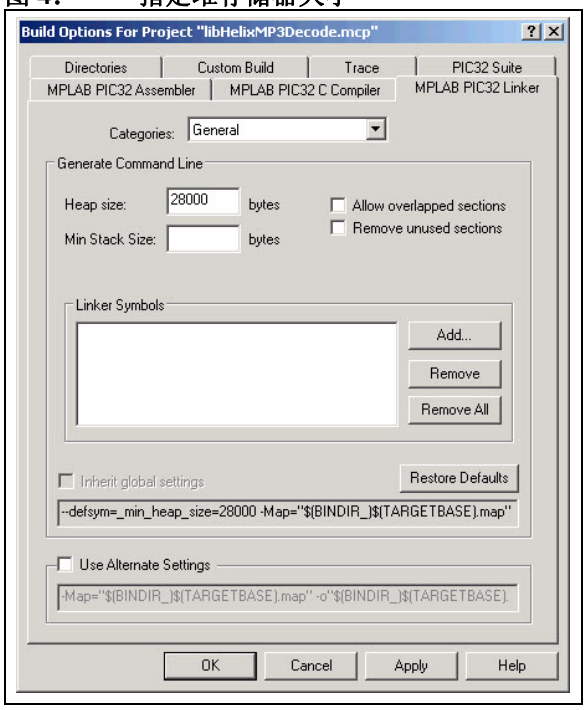
编译时，需要定义MIPS符号。这将编译适用于PIC32MX平台的 Helix MP3 解码器源代码。单击 ***Project>Build Options>Project***。从 **MPLAB PIC32 C Compiler**（MPLAB PIC32 C 编译器）选项卡中，将 MIPS 宏添加到编译流程，如图 3 中所示。

图 3： 将 MIPS 预处理器宏添加到编译选项



解码器需要堆存储器才能工作。不带输入和输出缓冲区的解码器所需的堆存储器大小是 28 KB，这需要在编译时指定。单击 *Project>Build Options>Project*，并选择 **MPLAB PIC32 Linker**（MPLAB PIC32 链接器）选项卡。指定堆大小，如图 4 中所示。

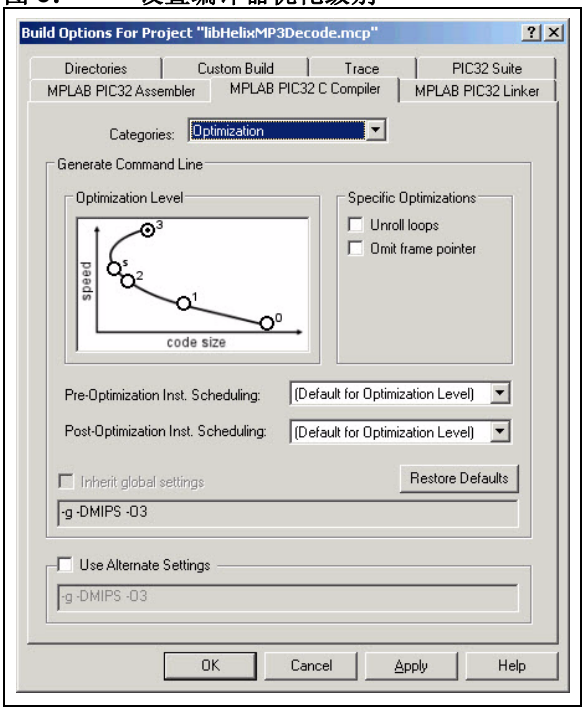
图 4：　　指定堆存储器大小



编译 Helix MP3 解码器源代码时，应将优化级别设置为 O3。单击 *Project>Build Options>Project*，并选择 **MPLAB PIC32 C Compiler** 选项卡。在 Categories （类别）字段的下拉菜单中选择 Optimization（优化）。将优化级别设置为 O3，如图 5 中所示。

Helix MP3 解码器源代码现在即可用于 PIC32MX 应用程序。

图 5：　　设置编译器优化级别



Helix MP3 解码器应用程序接口 （API）

Helix MP3 解码器 API 提供执行以下操作的函数：

- 初始化解码器
- 关闭解码器
- 检测帧同步
- 获取帧信息，如位速率和采样频率
- 解码 MP3 帧

> **注：** 演示应用程序使用 RTLL 技术，并通过系统加载程序接口间接调用 Helix MP3 解码器 API。因此，`MEB USB Thumb Drive MP3 C32 Demo.mcp` 源代码不能直接调用 Helix MP3 解码器 API。

使用 Helix MP3 解码器的第一步是在所有源文件中包含所需的头文件以调用解码器API。要包含的文件为`coder.h` 和 `mp3dec.h` （见例 3）。

例 3：

```
// 以下文件必须包含在
// 源代码中以调用
// Helix MP3 解码器 API

#include "coder.h"
#include "mp3dec.h"
```

Helix MP3 解码器是完全可重入的，这意味着解码器状态完全封装在一个数据结构中。 MP3InitDecoder() 函数为该数据结构分配存储器，将其初始化，然后返回一个指向初始化的数据结构的指针。此函数使用动态存储器分配，需要通过堆进行分配。如果此函数返回零值，则分配不成功 （见例 4）。

如果应用程序想要关闭解码器，则可以使用 MP3FreeDecoder() 函数释放分配给解码器的存储空间 （见例 5）。此函数可清除解码器占用的存储空间。

MP3 文件通常通过 ID3 标签存储附加信息。这些标签包含歌曲、艺术家、专辑和流派等相关信息。而提供给 Helix MP3 解码器函数的输入帧不应包含此信息。应用程序必须找到文件或者输入流中实际 MP3 数据的起始点。这可通过 MP3FindSyncWord() 函数实现。此函数接受一个输入缓冲区 （应该是从 MP3 文件读取的一部分字节），找到 MP3 帧头中的同步字，然后以距离输入缓冲区起始点的偏移量的形式返回此位置。同步字指示 MP3 帧的起始点。如果返回一个负值，则说明输入缓冲区未包含同步字或 MP3 帧起始点 （见例 6）。

例 4：

```
// 初始化 Helix MP3 解码器。
// 这将指向 MP3 解码器数据结构。

HMP3Decoder mp3Decoder;
mp3Decoder = MP3InitDecoder();
if(mp3Decoder == 0)
{
// 这意味着存储器分配失败。这通常在堆存储空间不足时发生。
// 请使用其他堆存储空间重新编译代码。

while (1);
}
```

例 5：

```
// 关闭解码器。

MP3FreeDecoder(mp3Decoder);

// 此时， MP3 解码器占用的存储空间被释放。
```

例 6：

```
// 在输入流中找到有效的 MP3 帧起始点

while(!endOfFile(mp3File))
{

        // 读取输入文件

        nRead = fread(mp3File,input,MAX_FRAME_SIZE);
        if(nRead == 0)
        {

        // 已到达文件的末尾，但没有找到有效的 MP3 帧起始点。
        // 执行一些操作。

        NotValidMP3File();
        break;
        }
        else
        {
            offset = MP3FindSyncWord(input,MAX_FRAME_SIZE);
            if(offset < 0)
            {

            // 输入缓冲区未包含帧起始点。读取另一帧。

            continue;
            }
            else
            {

            // 找到帧起始点。偏移量包含输入缓冲区内的
            // 帧起始点位置。

            foundStartOfFrame = TRUE;
            break;
            }
        }
    }
```

MP3FindSyncWord() 函数可能会找到同步字，但这可能不是实际的 MP3 帧起始点。可能存在 ID3 标签中某些数据与同步字匹配的情况。在这种情况下，如果将输入传递到 MP3 解码器函数，将返回一个错误。应用程序应相应地处理该错误。

本文档随附的 MP3 播放器应用程序设计用于处理立体声编码且采样率为 44.1 kHz 的 MP3 文件。Helix MP3 解码器提供了 MP3GetNextFrameInfo() 函数，用于返回尚未处理的 MP3 帧的音频属性。应用程序可根据需要使用此信息。MP3FindSyncWord() 函数用于定位帧起始点（Start of Frame，SOF），从而指示可能的 MP3 帧。然后可使用 MP3GetNextFrameInfo() 函数提取尚未解码的 MP3 帧的相关信息。该信息以一个 MP3FrameInfo 类型数据结构的形式返回。如果 MP3 帧无效，则返回一个错误（见例 7）。

MP3FrameInfo 数据结构中包含以下信息：

- 被处理帧的位速率
- 音频通道数（单声道为一个，立体声为两个）
- 编码音频采样率
- 每次采样的位数
- 音频采样中解码音频帧的大小（立体声采样视为两次音频采样）
- MPEG 层
- 层版本

MP3GetLastFrameInfo() 函数返回相同的信息，但返回的是已解码的音频帧的信息。例 8 显示了 MP3GetLastFrameInfo() 函数的使用方法。

例 7：

```
// 获取要解码的下一帧的相关信息。
// 前提是假定已使用 MP3FindSyncWord() 函数定位 MP3 帧起始点。

MP3FrameInfo frameInfo;

if(foundStartOfFrame == TRUE)
{
    int error;
    error = MP3GetNextFrameInfo(mp3Decoder, &frameInfo, input);
    if(error == MP3_INVALID_FRAME_HEADER)
    {
            // 这意味着 MP3FindSyncWord 函数已找到同步字，
            // 但并不是帧起始点。发生这种情况的原因可能是
            // 在 ID3 标签中找到了同步字。

            GetAnotherFrame();
    }

else if(frameInfo.sampRate != 44100)
    {
            // 在本例中，我们只需要采样率为 44100 Hz 的数据。
            // 因此忽略此帧。

            IgnoreThisFrame();
    }

}
```

例 8：

```
// 获取上一个解码帧的相关信息。假定该帧是
// 通过调用解码函数来解码的。

MP3FrameInfo mp3frameInfo;
int nOutputSamples;

MP3GetLastFrameInfo(mp3Decoder, &mp3FrameInfo);

// 获取输出原始音频帧的大小。

nOutputSamples = mp3FrameInfo.outputSamps;
```

MP3Decode() 函数用于解码已编码的 MP3 帧。此函数调用核心 MP3 解码算法。一次解码操作可能无法处理整个输入编码帧。在这种情况下，输入指针将前移指向未处理字节的起始点。解码器函数同时返回总的未处理字节数。 MP3 应用程序可利用此信息来准备输入缓冲区。为使操作正确，新数据应附加到未处理字节的末尾。

对于独立的 MP3 帧，可将输入帧格式化成标准 MPEG 流。在 MP3 播放器示例中已经使用了标准 MPEG 流选项。已解码原始数据的格式取决于音频通道数。对于单声道音频数据，每个音频输出采样都是唯一的。对于立体声音频数据，输出音频采样组织成左声道和右声道交错 （LRLRLR…）的音频采样。解码函数返回一个错误值指示解码过程的结果。MP3 应用程序可根据错误类型采取适当操作 （见例 9）。

例 9：

```
// 解码 MP3 帧。
// 假设使用了 MP3FindSyncWord() 函数查找帧起始点。

short output[MAX_NCHAN * 1152];
int err;
int bytesLeft

bytesLeft = INPUT_BUF_SIZE;
err = MP3Decode(mp3Decoder, &input, &bytesLeft, output, 0);

// bytesLeft 为输入缓冲区中剩余的字节数。
// 输入缓冲区将指向第一个未处理字节。

// 此代码示例说明如何处理错误。
// 不同应用程序的处理方法可能不同。

switch(err)
{
        case ERR_MP3_INDATA_UNDERFLOW:
                CloseMP3File();
                break;
        case ERR_MP3_MAINDATA_UNDERFLOW:
                ReadMoreMp3Data();
                break;
        case ERR_MP3_FREE_BITRATE_SYNC:
                CloseMP3File();
                break;
        default:
                CloseMP3File();
                break;
}

// MP3GetLastFrameInfo() 函数可用于获得有关帧的信息。
// 以下是一个示例。

MP3GetLastFrameInfo(mp3Decoder, &mp3FrameInfo);
if(mp3FrameInfo.outputSamps != 0)
{
        // 向输出 DAC 写入数据

WriteDataToDAC(output, mp3FrameInfo.,outputSamps);
}
```

表 1 列出了在 PIC32MX 单片机上运行 Helix MP3 解码器时的存储器要求。表 2 列出了计算要求。

表 1： HELIX MP3 解码器存储器要求

存储器类型	大小（字节）	备注
程序存储器	53000	—
数据存储器	28000	仅解码器需要。
输入缓冲区	1940	最大 MP3 帧大小。
输出缓冲区	2304	处理立体声音频数据时，输出缓冲区需要的最大大小。

表 2： HELIX MP3 解码器计算要求

函数	MIPS	备注
`MP3Decode()`	26	处理器时钟为 80 MHz，采用 O3 优化选项编译代码所需的计算速度。

运行时库加载

应用程序开发人员可能会考虑在应用程序中使用开源代码组件。在这种情况下，开源代码许可可能要求开源代码许可的条款和条件涵盖专有应用程序代码。这种要求可能会给程序开发人员或所有者带来不便。针对这种情况，本源代码随附的 MP3 应用示例采用了一种技术，其中：

- 开源代码不与主应用程序源代码链接。两种代码分别编译，并且不相互链接。
- 开源代码库在运行时通过加载程序以一组函数指针的形式加载。

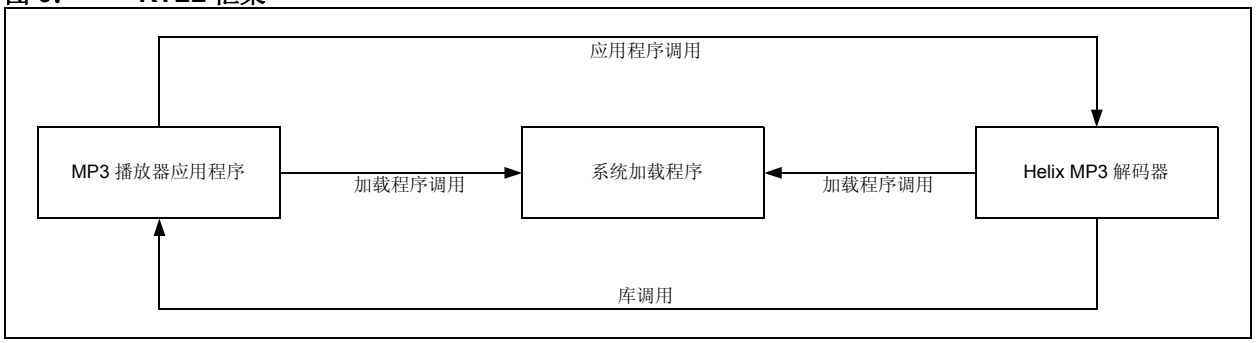
这种技术称为运行时库加载（RTLL）。MP3 播放器应用程序使用RTLL技术在运行时加载Helix MP3解码器。图 6 显示了 RTLL 操作的概念框图。

RTLL 技术使用系统加载程序来加载 Helix MP3 解码器。加载过程包括初始化 RAM、将表从程序存储器复制到 RAM（如果需要），以及获得一组指向解码器导出的函数的指针。MP3 播放器应用程序通过系统加载程序导出其函数，这些函数可由 Helix MP3 解码器调用。同样，通过系统加载程序将代码添加到 MP3 解码模块以公开一组函数，这些函数可由主应用程序调用。

MP3 播放器应用程序包含两个项目，创建两个不同的 hex 文件（程序映像）。一个文件用于主应用程序（也包含 USB 和图形协议栈），另一个文件用于 Helix MP3 解码器。这两个程序映像在器件程序存储器的不同单元进行编程。MP3 应用程序从而使用 RTLL 技术访问 Helix MP3 解码器。此技术需要对 C32 链接器使用的默认链接器脚本进行修改。程序闪存和数据 RAM 部分的起始位置需要进行调整，以实现 RTLL 技术。对于 Helix MP3 解码器，应将这些部分放在相应存储器区域的末尾。程序存储器区域设置为从距离程序存储器末尾的 72 KB 处开始，数据 RAM 设置为从距离 KSEG1 RAM 末尾的 256 字节处开始（堆栈存储区对于 Helix 来说已足够）。同样，创建 MP3 播放器程序映像时使用的链接器脚本的大小也将进行调整，以适应 Helix MP3 解码器。程序存储器减少 72 KB， KSEG1 RAM 减少 256 字节。

在本讨论的其余部分中，术语“主应用程序”指 MP3 播放器应用程序，术语“库”指 Helix MP3 解码器。两者统称为模块。 RTLL 技术使用动态模块头数据结构 `module_dyn_hdr` 来保存有关模块的信息。主应用程序、系统加载程序和库具有各自的动态模块头。例 10 显示了 `module_dyn_hdr` 的定义。

图 6：　RTLL 框架



例 10：

```
typedef struct
{
        // 模块签名
        const char module_sign[_MCHP_DYN_HDR_SIGNATURE_SIZE_];

        // 模块的名称
        const char module_name[_MCHP_MODULE_NAME_SIZE_];

        // 指向模块初始化的指针
        const module_init_dcpt* module_init;

        // 指向导出描述符的指针
        const export_dcpt*      module_export_tbl;

        // 指向导入描述符的指针
        const import_dcpt*      module_import_tbl;

        // 指向模块数据的指针
        module_data_dcpt*       module_data;

        // 模块信息
        const module_info_dcpt  module_info;

}module_dyn_hdr;
```

每个模块的动态头数据结构都包含有关导入和导出的函数的信息。导入的函数包含在模块导入表（module_import_tbl）中。导出的函数包含在模块导出表（module_export_tbl）中。例 11 显示了如何初始化主应用程序模块的模块动态头。

例 11：

```
const module_dyn_hdr __attribute__((__section__(".Module_Header_Section"))) _ModuleLoadHdr =
{
        _MCHP_DYN_HDR_SIGNATURE_,           // 签名
        MAIN_SERVICES_LIB,                  // 库名称
        0,                                  // 模块初始化
        &SystemExports,                     // 导出
        &_DefaultModuleImports,             // 导入
        &module_data,                       // 专有数据
        _DefaultModuleInfo(0x0100),         // 模块信息

};
```

主模块导出 Helix MP3 库所需的 malloc() 和 free() 函数。主应用程序模块不导入任何其他模块函数，_DefaultModuleImports 表为空。RTLL 技术允许系统加载程序以模块的形式加载。但是，在本特定示例中未采用此方式加载系统加载程序，而是将其静态链接到主应用程序模块。在编译主应用程序模块时，通过指定 LOADER_STATIC_LINK 选项强制执行此选择。

主应用程序调用 dlopen() 函数 （位于 AudioUSBTasks.c 文件的 AudioUSBInit() 函数中）初始化 RTLL 系统 （见例 12）。

dlopen() 函数加载 p_modules[] 数组中指定的所有模块。加载过程将调用与各个模块关联的启动代码（如果有的话）。 Helix MP3 库模块使用改进的 C 运行时启动代码来初始化其存储器。dlopen() 函数然后返回一个指向 Helix MP3 库模块动态头的指针。

主应用程序则使用 dlsym() 函数获得一个 Helix MP3 库模块入口点的句柄 （见例 13）。

这将返回一个指向 helix_lib.c 文件中定义的 HelixLibEntry() 函数的指针。此函数根据指定的参数调用 Helix MP3 解码器 API。例 14 显示了部分的 HelixLibEntry() 函数实现。

HelixLibEntry() 函数的 fCode 参数决定要调用的 Helix 库函数。例如，如果 fCode 为 0，则调用 MP3InitDecoder() 函数。应该注意的是，主应用程序模块通过系统加载程序 dlsym() 调用来获得 HelixLibEntry() 函数的句柄（MP3DecoderFunctions）。

主应用程序模块现在使用 MP3DecoderFunctions 句柄调用 Helix MP3 解码器 API 函数。例 15 显示了主应用程序模块通过 MP3DecoderFunctions 句柄调用 MP3Decode() 函数的过程。

例 12：

```
/* 获得 MP3 解码器的句柄并获取解码器库函数的
 * 入口点。 */

hMP3DecoderLibrary = dlopen("Helix Library", 0)
```

例 13：

```
MP3DecoderFunctions = (void* (*)(int, int, ...))dlsym(
hMP3DecoderLibrary, "HelixLibEntry");
```

例 14：

```
void* HelixLibEntry(int fCode, int nparams, va_list args)
{
        void*   res = (void*)-1;

        //库入口点
        switch(fCode)
        {
            case 0:
                if(nparams == 0)
                {
                    res = MP3InitDecoder();
                }
                break;

            case 1:
            if(nparams == 1)
            {
                MP3FreeDecoder(va_arg(args, HMP3Decoder));
                res= 0;
            }
            break;
```

例 15：

```
// MP3_DECODE_FUNCTION 为 MP3Decode() 函数的 fCode 值。
// 第二个参数指示后面的参数总数。

err = (int)(*MP3DecoderFunctions)(MP3_DECODE_FUNCTION,5,hMP3Decoder, &readPtr, &bytesLeft,
outBuf, 0);
```

MP3 播放器应用程序

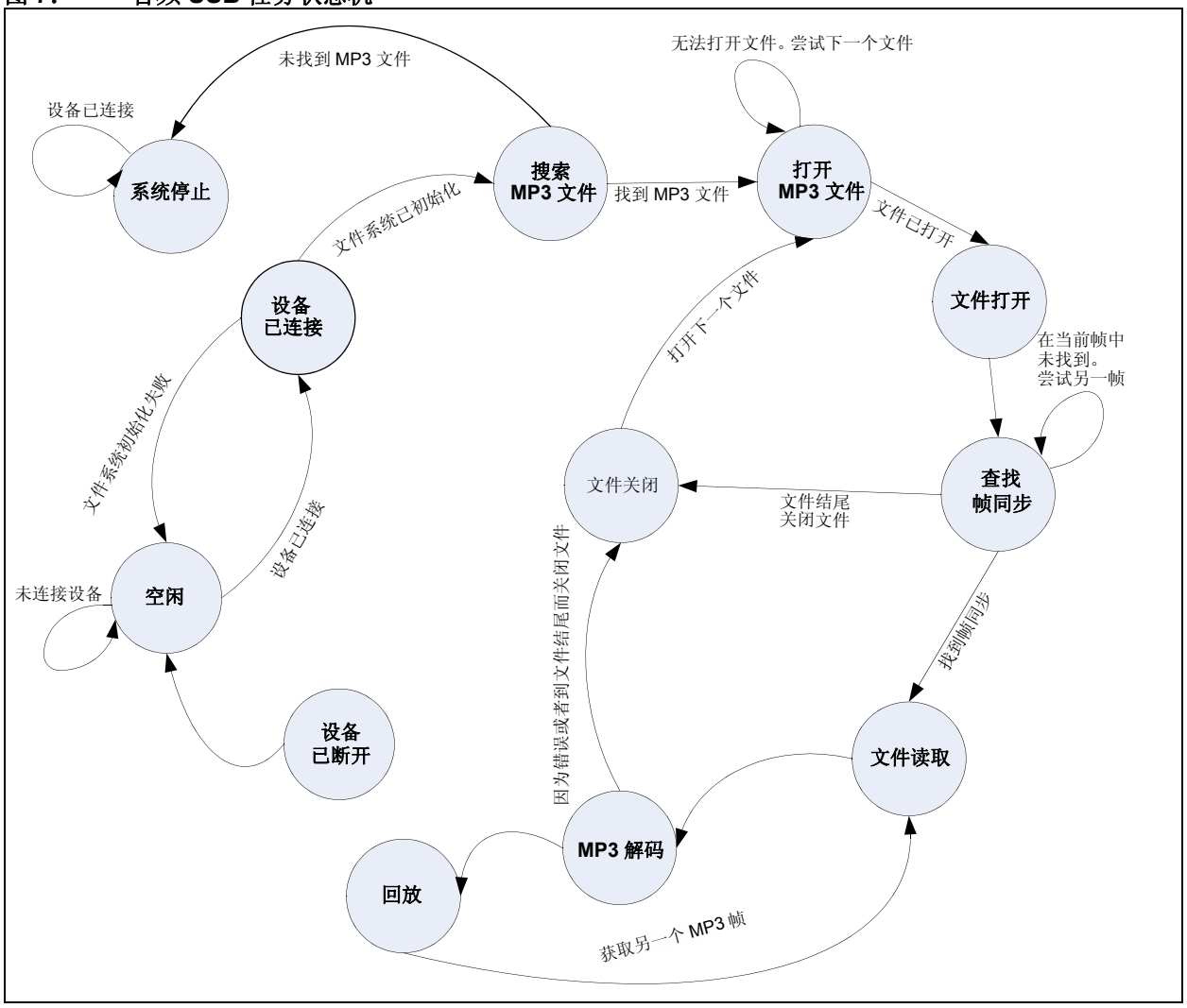
MP3 播放器应用程序使用 Helix MP3 解码器（通过 RTLL）、Microchip 的 USB 协议栈和图形协议栈实现 MP3 播放器。该应用程序使用 USB 协议栈的大容量存储设备（Mass Storage Device，MSD）主机组件读取 U 盘。图形协议栈提供触摸屏和用户界面功能。

关于 USB 和图形协议栈操作有完备的文档，本应用笔记不对此进行讨论。MP3 播放器的内核逻辑以状态机的形式在`AudioUSBTasks.c`文件中的`AudioUSBTasks()`函数中实现。图 7 显示了此状态机的图形表示。

当处于空闲状态时，状态机等待 U 盘连接。设备连接后，代码尝试初始化 U 盘文件系统。应用程序将搜索 U 盘上的 MP3 文件。如果找到文件，则将文件名存储在`fileNames[]`数组中。最多可存储`MAX_FILES`个文件名。显示屏上的文件列表框将更新文件名。同时打开第一个文件，代码将搜索第一个帧起始点。如果找到帧起始点，将从文件读取该帧的其余部分并传递给解码函数。已解码的音频帧发送到 WM8731 音频 DAC 器件进行回放。文件读取和解码操作将继续，直到文件结尾；然后将关闭该文件并打开下一个文件。

`currentPlaybackFileIndex`变量以`fileNames[]`数组中索引的形式记录当前正在处理的文件。作为对 **Next**（下一个）或 **Previous**（前一个）按钮动作的响应，应用程序将更改此变量并跳转到“打开 MP3 文件”状态。

图 7：　音频 USB 任务状态机



运行演示应用程序

本应用程序笔记随附MP3播放器演示应用程序源代码。该 MP3 播放器应用程序可对以 44.1 kHz 采样率编码的立体声 MP3 文件进行解码。以其他采样率编码或者单声道编码的 MP3 文件将被忽略。这纯粹是一个应用程序设计选择，而不是 Helix MP3 解码器的限制。该解码器支持符合 MP3 格式规定的全部位速率和采样频率。

MP3 播放器应用程序设计为在多媒体扩展板（Multimedia Expansion Board， MEB）（部件编号：DM320005）上运行，连同 PIC32MX USB 入门工具包 II（部件编号：DM320003-2）或 PIC32 以太网入门工具包（部件编号：DM320004）。MEB 提供 24 位音频 ADC/DAC、触摸屏显示，并通过入门工具包连接器支持 PIC32MX 器件。有关详细信息，请访问多媒体扩展板网页 www.microchip.com/MEB。

PIC32MX USB 入门工具包 II 和 PIC32 以太网入门工具包使用 PIC32MX795F512L 器件。该器件提供 512 KB 的程序闪存和 128 KB 的数据 RAM。该器件还提供 USB 模块。入门工具包包括 USB 主机和设备连接器。

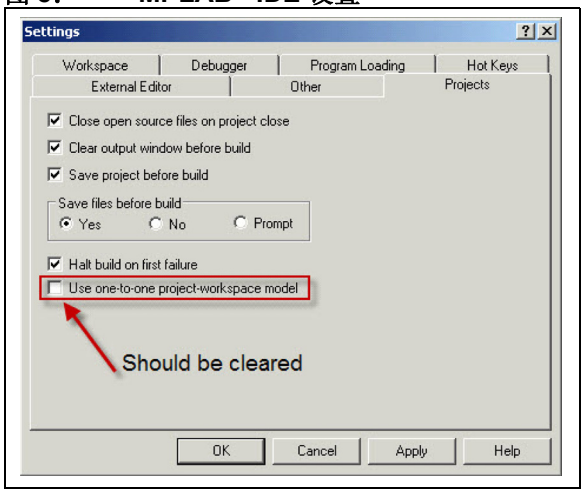
需要一副耳机和一个包含 MP3 文件的 U 盘来测试演示源代码。 MP3 文件必须放在 U 盘的根目录下。

编译和烧写演示应用程序

本部分介绍编译 MP3 播放器演示程序和烧写 PIC32MX 器件需要的步骤。以下为使用 PIC32MX USB 入门工具包 II 时的步骤。使用 PIC32 以太网入门工具包时，应使用 `MEB ENET USB Thumb Drive MP3 Demo.mcw` MPLAB IDE 工作区文件。对于两种开发板来说，编译和运行演示的操作说明是相同的。这里假定用户已熟悉 MPLAB® IDE。

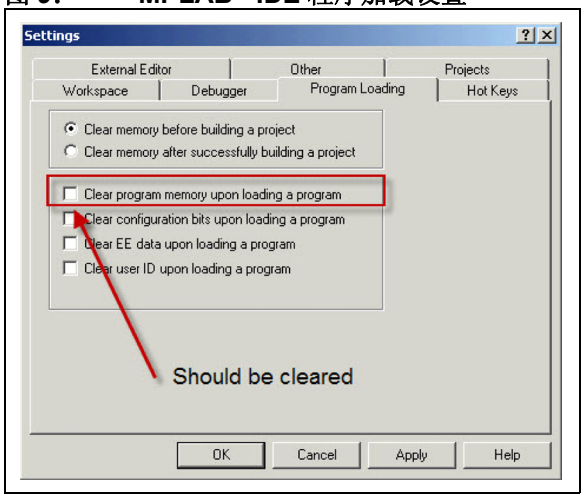
1. 打开 MPLAB IDE。单击 *Configure>Settings*（配置 > 设置）。单击 **Projects**（项目）选项卡并清除“Use one-to-one project-workspace model”（使用一对一项目-工作区模型）选项，如图 8 中所示。 单击 **OK**（确定）。

图 8： MPLAB® IDE 设置



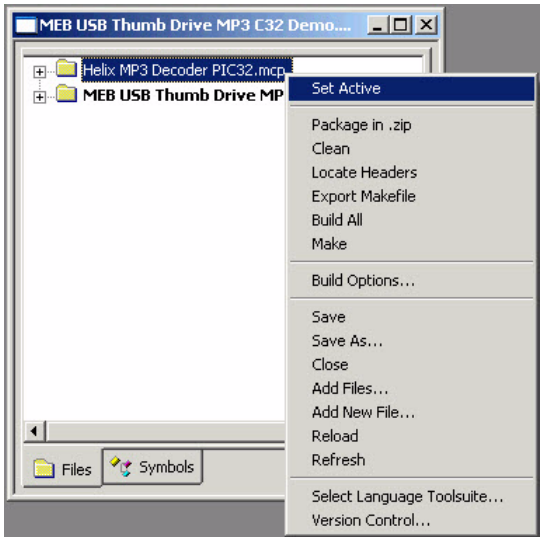
2. 单击 *Configure>Settings*。单击 **Program Loading**（程序加载）选项卡。清除 “Clear program memory upon loading a program”（加载程序时清除程序存储器）选项，如图 9 中所示。

图 9： MPLAB® IDE 程序加载设置



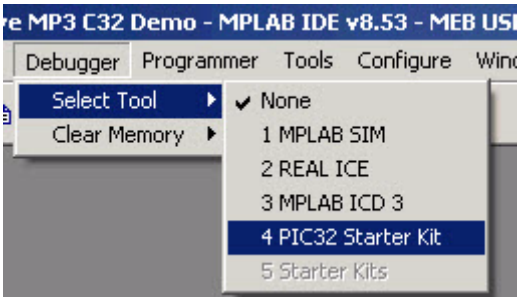
3. 单击 *File>Open Workspace*（文件 > 打开工作区），然后打开 `MEB USB Thumb Drive MP3 C32 Demo.mcw` MPLAB IDE 工作区文件。应用程序项目即加载到 MPLAB IDE 中。
4. 项目资源管理器窗口在工作区内显示两个项目。`Helix MP3 Decoder PIC32.mcp` 项目由 Helix MP3 解码器源代码组成，该源代码针对在 PIC32MX 单片机上运行做了修改。
5. 右键单击 `Helix MP3 Decoder PIC32.mcp` 项目，然后从菜单中选择 Set Active（设置激活），如图 10 中所示。

图 10：　使项目激活



6. 确保未选择调试器或者编程器。Debug/Release（调试 / 发布）的选择不会影响操作，可以忽略。使用“Build All”（编译所有）选项编译并生成项目。
7. 右键单击 `MEB USB Thumb Drive MP3 C32 Demo.mcp`项目，然后从菜单中选择Set Active。
8. 将 PIC32 USB 入门工具包 II 连接到 MEB。使用紧固件以确保入门工具包牢固地锁定在插座上。
9. 将入门工具包调试USB端口与PC上可用的USB端口相连接。
10. 在 MPLAB IDE 中，单击 *Debugger>Select Tool> PIC32 Starter Kit*（调试器 > 选择工具 >PIC32 入门工具包）以加载 PIC32 入门工具包，如图 11 中所示。

图 11：　选择调试器



11. 编译并生成项目。
12. 烧写 PIC32MX 器件。 MPLAB IDE 将两个 hex 映像烧写到器件上：
    - `Helix MP3 Decoder PIC32.elf`
    - `MEB USB Thumb Drive MP3 C32 Demo.elf`
13. *此步骤为可选*。检查地址为 0x9D06E000 的程序存储器单元。在 MPLAB IDE 中，单击 *View> Memory*（视图 > 存储器），然后在存储器窗口中单击 **Data View** （数据视图）。在存储器窗口中单击右键，选择 Go To（转到），然后输入地址 0x9D06E000。将显示生成的存储器窗口，如图 12 中所示。字符串 “Helix Library”指示烧写操作成功。

演示应用程序现在可以使用。

图 12：　程序存储器视图

Memory

Address	Virtual	00	04	08	0C	ASCII
1D06_DF60		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DF70		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DF80		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DF90		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DFA0		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DFB0		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DFC0		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DFD0		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DFE0		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_DFF0		FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	....
1D06_E000		48434D40	4F4D5F50	454C5544	4145485F	@MCHP_MO DULE_HEA
1D06_E010		00524544	696C6548	694C2078	72617262	DER.Heli x Librar
1D06_E020		00000079	00000000	9D07A2EC	9D07A2E0	y....... ....
1D06_E030		9D07A2C4	A001FF08	00000000	33636970	....pic3
1D06_E040		00786D32	00000000	00000000	0100031B	2mx..... ....
1D06_E050		A001FF00	A001FF00	9D06E000	9D076E24	....$n..
1D06_E060		0B41C337	00000000	10800003	00000000	7.A..... ....
1D06_E070		0B41C2F7	00000000	03E00008	00000000	..A..... ....
1D06_E080		24A5FFFF	18A00011	00003021	240700FF	...$.... !0.....$
1D06_E090		10000004	240800E0	00C5502A	1140000B	.......$ *P....@.

Data View　Code View

运行演示应用程序

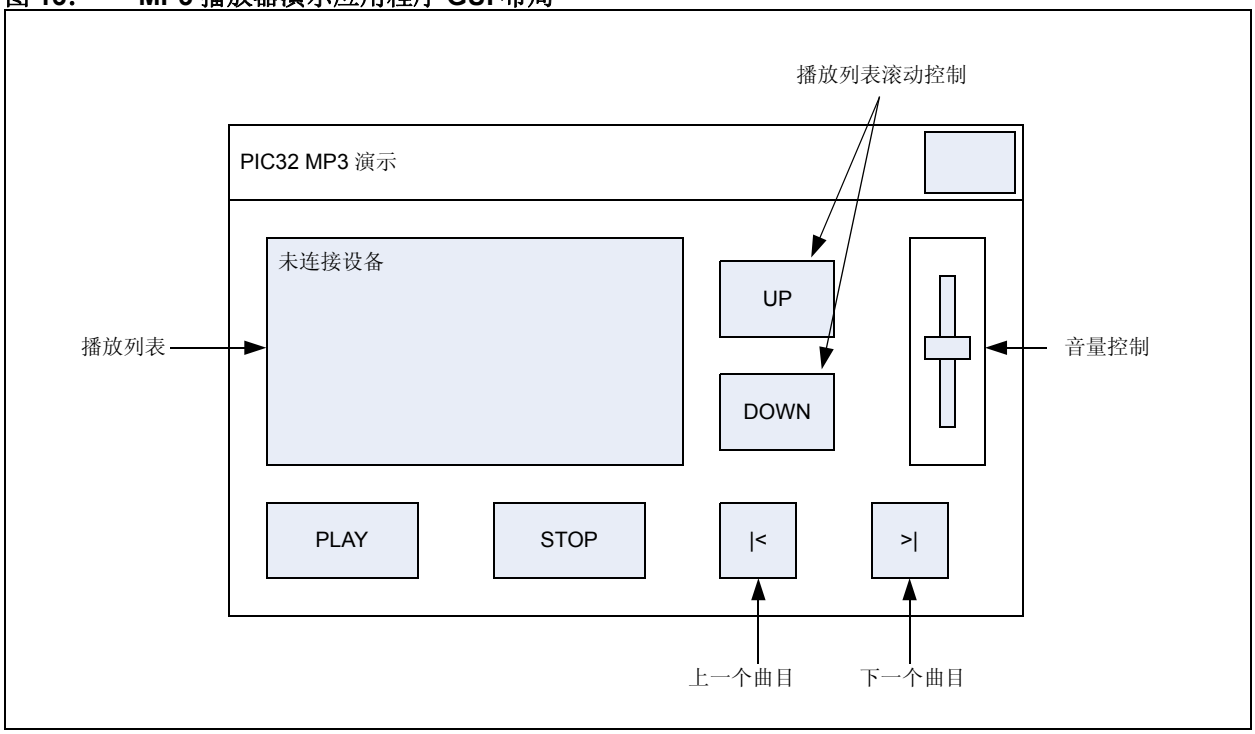
将耳机插入 MEB 上的耳机插孔中。在 MPLAB IDE 中，单击 **Run**（运行）。观察 MEB 触摸屏。图 13 说明了 MP3 播放器演示应用程序 GUI。

将 U 盘插入到 PIC32MX USB 入门工具包 II 上的 USB 主机插座中。播放列表将显示 U 盘根目录中的 MP3 文件。应用程序开始播放列表中的第一个 MP3 文件。

触摸屏按钮的功能如下：

- 音量控制滑动条用于调节音量。
- STOP （停止）按钮用于停止播放 MP3 文件。
- 回放时按下 PLAY （播放）按钮可暂停回放。再按一次则继续回放。
- 用户可使用 UP（向上）和 DOWN（向下）按钮浏览播放列表中的文件。触摸播放列表中的一个文件将选中该文件进行回放。

图 13： MP3 播放器演示应用程序 GUI 布局



项目文件

本应用笔记随附的源代码（见**附录 A：“源代码”**）包含适用于 PIC32MX USB 入门工具包 II 和 PIC32MX 以太网入门工具包的演示源代码。

使用 PIC32MX USB 入门工具包 II 时，在 MEB 上运行 MEB USB Thumb Drive MP3 Demo 文件夹中的应用程序。使用 PIC32MX 以太网入门工具包 II 时，在 MEB 上运行 MEB ENET USB Thumb Drive MP3 Demo 文件夹中的应用程序。

表 3 中提供的文件夹和文件描述适用于这两个应用程序文件夹。所有差异均已明确指出。

结论

Helix MP3 解码器是可以移植到 Microchip 32 位 PIC32MX 单片机上的开源 MP3 解码器。本应用笔记介绍了解码器 API，并提供 MP3 播放器应用程序以演示 MP3 解码器的使用。此外，还介绍并演示了 RTLL 技术。

表 3： 源代码文件和文件夹描述

项目名称	说明
h	包含 MEB USB Thumb Drive MP3 C32 Demo 项目所需的 Include 文件的文件夹。
lib	包含 USB 和图形协议栈归档的文件夹。
obj	用于存储临时板级支持包（Board Support Package，BSP）目标文件的文件夹。
src	包含 MEB USB Thumb Drive MP3 C32 Demo 项目所需的源文件的文件夹。
MP3 Decoder Source Code	包含 Helix MP3 解码器源代码的文件夹。
procdefs.ld	MEB USB Thumb Drive MP3 C32 Demo 项目和 Helix MP3 Decoder PIC32 MPLAB IDE 项目所需的经修改的链接器脚本文件。
MEB USB Thumb Drive MP3 C32 Demo.mcp 或 MEB ENET USB Thumb Drive MP3 Demo.mcp	MPLAB IDE 项目文件。
MEB USB Thumb Drive MP3 C32 Demo.mcw 或 MEB ENET USB Thumb Drive MP3 Demo.mcw	MPLAB IDE 工作区文件。
CleanUp.bat	用于清除临时文件的批处理文件。
MAL BSP Files Archive.bat	用于更新和生成 BSP 归档的批处理文件。
MAL GFX Stack Archive.bat	用于更新和生成图形协议栈归档的批处理文件。
MAL USB Stack Archive.bat	用于更新和生成 USB 协议栈归档的批处理文件。
Readme.txt	包含运行演示相关信息的文件。
Alternative Configurations	包含 MEB 图形显示所需的 Include 文件的文件夹。

附录 A： 源代码

本应用笔记中提及的所有软件都以单个 WinZip 归档文件的形式提供。可从 Microchip 公司网站下载此文件：

www.microchip.com

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

ISBN：978-1-61341-159-9

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax:1-678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 1-408-961-6444
Fax: 1-408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario, Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 – 杭州
Tel: 86-571-2819-3180
Fax: 86-571-2819-3189

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 – 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 – 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

亚太地区

台湾地区 – 高雄
Tel: 886-7-213-7830
Fax: 886-7-330-9305

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-6578-300
Fax: 886-3-6578-370

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

05/02/11