

Building a Resound Module: A Walk-Through

We'll create *Zero.rmod*, a simple module that sets every value in the selection of a sound to zero. It will only work on 16-bit sounds (to save space here), but *will* work with Intel machines.

Create a Project

Start creating the folder *Zero* wherever convenient. This will be the project folder for Zero. Next, launch ProjectBuilder.app and choose *Project:New...* Locate your Zero folder. Change the Project Type to *Bundle*, and save the project in the Zero folder.

Adding Resound Resources

In the directory, copy the four files that come with the Resound distribution: *Module.h*, *ModuleMenuNode.h*, *ModuleSound.h*, and *ModuleProtocol.h*.

Add Some Code

Launch Edit.app, and create a new file. In this file, add the following:

```
#import "Module.h"

@interface ZeroModule:Module
{
}

- init;
- zeroSound:sender;

@end
```

Save this as *ZeroModule.h* in your Zero folder. Create another file with the following:

```
#import "ZeroModule.h"
#import <soundkit/soundkit.h>
#import <appkit/appkit.h>
```

```

@implementation ZeroModule

-initWith
{
    id returnVal=[super init];

    ModuleMenuNode* zeroNode=[[ModuleMenuNode alloc] init];

    [TheModuleMenuNode setLength:1];
    [TheModuleMenuNode setSubmenu: 0 : zeroNode];

    [zeroNode setName: "Zero"];
    [zeroNode setReceiver: self];
    [zeroNode setMessage:@selector(zeroSound:)];
    [zeroNode setLength:0];

    return returnVal;
}

```

Before we go on, this requires some explaining. *Module* is the class from which all modules subclass. It contains the machinery necessary to load your module into Resound. *ModuleMenuNode* is a special object with which you specify the submenu to build in Resound's Modules menu. This node is a tree-node, and every menu entry and submenu you add is added as a leaf in the tree.

The root of this tree is *TheModuleMenuNode*, which you've inherited from *Module*. This node represents the Modules menu itself. If you want, say, three items in the Modules menu, you set *TheModuleMenuNode*'s length to 3 and set submenus 0, 1, and 2 to three distinct module menu nodes you've created. As it is, we're only doing one node: *zeroNode*.

zeroNode is a leaf node in the tree, so we declare its name, its receiver (this instance of *ZeroModule*), and the message sent when choosing the menu item (**zeroSound:**). We've set the length to 0 to indicate that *zeroNode* is a leaf node.

Now add to this file:

```

- zeroSound:sender
{
    Sound* current=          [TheModuleController currentSound];
    SoundView* cview=        [TheModuleController currentSoundView];

    int dataFormat=          [current dataFormat];
    int channelCount=        [current channelCount];
    int sampleCount=         [current sampleCount];
    signed short* data;
    int x;

    int firstSample,sampleLength;

    [cview getSelection: &firstSample size: &sampleLength];

    if (current==NULL||!sampleLength)    // no sound
    {
        NXRunAlertPanel("No Sound",
            "There is no sound or selection with which to perform this operation." ,
            "Okay",NULL,NULL);
        return NULL;
    }

    if (dataFormat!=SND_FORMAT_LINEAR_16)
    {
        NXRunAlertPanel("Wrong Sound Format",
            "This Operation only works with 16-bit linear sounds." ,
            "Okay",NULL,NULL);
        return NULL;
    }
}

```

TheModuleController, which we've inherited from *Module*, provides us access to Resound's facilities. Above we've grabbed the current sound and soundview as displayed by Resound, and have begun querying the sound and soundview. If there isn't a selection or the sound is NULL, we pop up a message telling the user that

there's no sound, and exit. Likewise if the sound is not 16-bit.

Note that **getSelection:** does not always return proper values in NeXT's sound view (a bug). Resound's customized sound view does not fix this entirely, but *does* at least guarantee that the selection will be between 0 and the maximum number of samples in the sound (NeXT's is often way out of range). Continuing:

```
[TheModuleController stop];
[current compactSamples];
data = (signed short*)[current data];
SNDSwapSoundToHost
    ((void*)data, (void*)data, sampleCount, channelCount, dataFormat);
```

We stop any sound currently playing (just in case), compact the samples in the sound so we have a clean array on which to operate, and swap the data in the sound to the format that our type of computer can read (Intel machines' data is backwards from most other machines). Note that we set the data pointer only *after* compactSamples has been performed, since compactSamples changes the pointer. Onward:

```
for (x=firstSample;x<firstSample+sampleLength;x++) data[x]=0;

SNDSwapHostToSound
    ((void*)data, (void*)data, sampleCount, channelCount, dataFormat);
[TheModuleController soundChanged];
return self;
}

@end
```

We finish by actually zeroing out the array between firstSample and firstSample+sampleLength, then swapping the sound back to its "proper" format, and finally informing Resound that we've made modifications to the sound itself.

Compiling the Code

Return to ProjectBuilder.app. Add *ZeroModule.m* as a class in your Zero project by choosing the "Files" button, then double-clicking on "Classes". *ZeroModule.h*, if it's also been saved in the Zero project directory, should be automatically added under "Headers". If not, add it. Finally, add the three files *Module.h*,

ModuleMenuNode.h, and *ModuleProtocol.h* as headers in the project by double-clicking "Headers".

Click on the button "Builder". Make certain the target is *Bundle*. By clicking on *Options...*, choose the architectures you want to build for (if you're distributing a module, you should build it for as many architectures as possible—for this example, just pick the architecture of your current machine).

Lastly, Click on "Build" to save and build the project. A bundle called *Zero.bundle* should appear in the Zero project directory. Move this bundle into the directory */LocalLibrary/Resound* or into the Resound subdirectory of your home library directory (*~/Library/Resound*) so Resound will find it. Rename the bundle *Zero.rmod*. "rmod" is the standard Resound module extension.

Try It Out

Launch *Resound.app*. Record a sound and convert it to 16-bit linear. Select a section of the sound. Choose *Modules:Zero* and have fun!

Cleaning Up

Before you distribute your object to others, you should strip it of extraneous symbols (which should make the bundle considerably smaller). To do this, open the file *Makefile.postamble* in your Zero project directory. Change the line

```
#RELOCATABLE_STRIP_OPTS = -x -u
```

To

```
RELOCATABLE_STRIP_OPTS = -x -u
```

If you're moving *Zero.bundle* to */LocalLibrary/Resound*, add (to the beginning of the file) the line

```
INSTALLDIR = /LocalLibrary/Resound
```

Otherwise, if you're moving *Zero.bundle* to *~/Library/Resound*, add (to the beginning of the file) the line

```
INSTALLDIR = ~/Library/Resound
```

Save the file and re-build the project with a target of *Install*. The bundle *Zero.bundle* will appear in your library

directory (depending on what you picked above). Change its name to *Zero.rmod*.