# Dynamic Range Compressor Module Documentation

Module: compressor.c

## 1. Overview

This document provides a detailed description of the dynamic range compressor audio filter. The module was originally a single-band dynamic range compressor based on the LADSPA SWH plugins. It has been significantly updated to include a full-featured, three-band compressor, providing much more granular control over audio dynamics.

The module now operates in one of two modes:

1. **Single-Band Compressor**: The original legacy mode that applies compression uniformly across the entire frequency spectrum.
2. **Multi-band Compressor**: A new, advanced mode that splits the audio signal into three frequency bands (Low, Mid, High) and compresses each one independently.

## 2. Key Enhancements

The transition from `compressor.txt` (the original code) to `new-updated-compressor.c` introduces several major enhancements:

- **Dual-Mode Architecture**: The filter is no longer monolithic. It has been refactored to support both single-band and multi-band operation through separate initialization paths and processing loops.
- **Three-Band Compression**: The core of the update is a new three-band compressor with the following features:
    - **Crossover Filters**: The audio is split using two 2nd-order Linkwitz-Riley crossover filters, ensuring a smooth frequency response across the bands. The crossover frequencies are configurable.
    - **Independent Band Processing**: Each band (Low, Mid, High) has its own set of compression parameters: Threshold, Ratio, Attack, Release, and Makeup Gain.
    - **Dedicated Configuration**: New configuration options and callbacks have been added to VLC's interface to control the multi-band parameters.

- **Code Refactoring for Clarity**: The codebase has been restructured to separate the logic for single-band and multi-band operations, improving readability and maintainability. Common helper functions are shared between the two modes.

## 3. How the Compressor Works Now

### 3.1. Single-Band Mode

This mode remains functionally identical to the original implementation. It analyzes the level of the incoming audio signal (a mix of RMS and Peak levels) and compares it to a threshold. If the level exceeds the threshold, the signal's gain is reduced according to the specified ratio. This provides effective, general-purpose dynamic range control.

### 3.2. Multi-band Mode

The multi-band mode offers a more surgical approach to compression. The process for each audio sample is as follows:

1. **Frequency Splitting**: The incoming audio sample is first fed into a crossover network.
    - The first crossover splits the signal into a low band and a combined mid/high band.
    - The combined mid/high signal is then fed into a second crossover, which separates it into the final mid and high bands.
    - This results in three separate audio streams: **Low**, **Mid**, and **High**.
2. **Independent Compression**: Each of the three bands is then processed by its own dedicated compressor instance.
    - The level of the **Low** band is analyzed, and gain reduction is applied based on the "Low Band" settings (threshold, ratio, etc.).
    - Simultaneously, the **Mid** and **High** bands are compressed using their respective settings.
    - This allows, for example, for heavy compression on bass frequencies to control "boominess" without affecting the clarity of vocals in the mid-range or the sparkle of cymbals in the high-range.
3. **Signal Recombination**: After each band has been independently compressed, the three audio streams are summed back together into a single stream, which becomes the final output of the filter.

## 4. Detailed Implementation Description

### 4.1. Code Structure and Refactoring

The implementation was refactored to support the dual-mode functionality.

- **System Structures**:
  - `filter_sys_t` was removed.
  - `singleband_sys_t`: A new structure that holds all state and parameters exclusively for the single-band mode.
  - `multiband_sys_t`: A new structure for the multi-band mode. It contains an array of `compressor_band_t` structures (one for each band), the crossover filter states, and shared lookup tables.
- **Module Registration**: The `vlc_module_begin()` block now registers two distinct audio filters under the same module:
  - The primary "Compressor" which uses the single-band implementation (`OpenSingle`, `CloseSingle`, `DoWorkSingle`).
  - A "MB Compressor" submodule which uses the multi-band implementation (`OpenMultiband`, `CloseMultiband`, `DoWorkMultiband`).
- **Shared Functions**: Utility functions common to both modes (e.g., `DbInit`, `Db2Lin`, `Lin2Db`, `Clamp`, `RmsEnvProcess`) have been preserved and are called by both processing paths.

### 4.2. Key Functions and Data Structures

| Function/Structure | Purpose |
| --- | --- |
| `singleband_sys_t` | Holds all state for the single-band compressor, including its single `compressor_band_t` and parameters. |
| `multiband_sys_t` | Holds all state for the multi-band compressor. This includes an array of `compressor_band_t` (one for each band), crossover filter coefficients and state, and shared lookup tables. |

| | |
|---|---|
| `compressor_band_t` | A generic structure that contains the full set of compressor parameters and state variables (e.g., `f_threshold`, `f_ratio`, `f_env_peak`) for a single compressor instance. It is used in both single-band and multi-band modes. |
| `biquad_coeffs` | Stores the coefficients for the biquad filters used to create the crossovers. |
| `OpenSingle` / `OpenMultiband` | Initialization functions. They allocate memory for their respective system structures, initialize parameters from stored settings, create the necessary callbacks, and set up the filter operations. `OpenMultiband` also calculates the initial crossover filter coefficients. |
| `DoWorkSingle` / `DoWorkMultiband` | The main processing loops for their respective modes. `DoWorkSingle` applies compression to the full-band signal, while `DoWorkMultiband` orchestrates the splitting, independent compression, and recombination of the audio bands. |
| `CloseSingle` / `CloseMultiband` | De-initialization functions. They remove parameter callbacks and free all allocated memory. |
| `calculate_crossover_coeffs` | Calculates the biquad filter coefficients for the Linkwitz-Riley low-pass and high-pass filters based on the sample rate and desired crossover frequency. |
| `CrossoverProcess` | The core of the signal splitting. It takes a single input sample and runs it through the two stages of biquad filters to produce the low, mid, and high band outputs. |

| | |
|---|---|
| `CompressBand` | The compression logic for a single frequency band. It performs envelope detection and applies gain reduction and makeup gain according to that band's unique settings. |
| `*Callback` Functions | The code contains two sets of callback functions. The original set (`AttackCallback`, `RatioCallback`, etc.) now updates the `singleband_sys_t`. A new, extensive set of callbacks (`MultibandLowThresholdCallback`, `MultibandMidRatioCallback`, etc.) has been added to handle real-time parameter updates for each of the three bands in the multi-band compressor. |