Combos

The Combo feature is a chording type solution for adding custom actions. It lets you hit multiple keys at once and produce a different effect. For instance, hitting `A` and `S` within the tapping term would hit `ESC` instead, or have it perform even more complex tasks.

To enable this feature, you need to add `COMBO_ENABLE = yes` to your `rules.mk`.

Additionally, in your `config.h`, you'll need to specify the number of combos that you'll be using, by adding `#define COMBO_COUNT 1` (replacing 1 with the number that you're using).

Also, by default, the tapping term for the Combos is set to the same value as `TAPPING_TERM` (200 by default on most boards). But you can specify a different value by defining it in your `config.h`. For instance: `#define COMBO_TERM 300` would set the time out period for combos to 300ms.

Then, your `keymap.c` file, you'll need to define a sequence of keys, terminated with `COMBO_END`, and a structure to list the combination of keys, and it's resulting action.

```
const uint16_t PROGMEM test_combo[] = {KC_A, KC_B, COMBO_END};
combo_t key_combos[COMBO_COUNT] = {COMBO(test_combo, KC_ESC)};
```

This will send "Escape" if you hit the A and B keys.

!> This method only supports basic keycodes. See the examples for more control.

Examples

If you want to add a list, then you'd use something like this:

```
enum combos {
  AB_ESC,
  JK_TAB
};

const uint16_t PROGMEM ab_combo[] = {KC_A, KC_B, COMBO_END};
const uint16_t PROGMEM jk_combo[] = {KC_J, KC_K, COMBO_END};

combo_t key_combos[COMBO_COUNT] = {
  [AB_ESC] = COMBO(ab_combo, KC_ESC),
  [JK_TAB] = COMBO(jk_combo, KC_TAB)
};
```

For a more complicated implementation, you can use the `process_combo_event` function to add custom handling.

```
enum combo_events {
  ZC_COPY,
```

```
  XV_PASTE
};

const uint16_t PROGMEM copy_combo[] = {KC_Z, KC_C, COMBO_END};
const uint16_t PROGMEM paste_combo[] = {KC_X, KC_V, COMBO_END};

combo_t key_combos[COMBO_COUNT] = {
  [ZC_COPY] = COMBO_ACTION(copy_combo),
  [XV_PASTE] = COMBO_ACTION(paste_combo),
};

void process_combo_event(uint8_t combo_index, bool pressed) {
  switch(combo_index) {
    case ZC_COPY:
      if (pressed) {
        tap_code16(LCTL(KC_C));
      }
      break;
    case XV_PASTE:
      if (pressed) {
        tap_code16(LCTL(KC_V));
      }
      break;
  }
}
```

This will send Ctrl+C if you hit Z and C, and Ctrl+V if you hit X and V. But you could change this to do stuff like change layers, play sounds, or change settings.

Additional Configuration

If you're using long combos, or even longer combos, you may run into issues with this, as the structure may not be large enough to accommodate what you're doing.

In this case, you can add either `#define EXTRA_LONG_COMBOS` or `#define EXTRA_EXTRA_LONG_COMBOS` in your `config.h` file.

You may also be able to enable action keys by defining `COMBO_ALLOW_ACTION_KEYS`.

Keycodes

You can enable, disable and toggle the Combo feature on the fly. This is useful if you need to disable them temporarily, such as for a game.

Keycode	Description

Keycode	Description
`CMB_ON`	Turns on Combo feature
`CMB_OFF`	Turns off Combo feature
`CMB_TOG`	Toggles Combo feature on and off

User callbacks

In addition to the keycodes, there are a few functions that you can use to set the status, or check it:

Function	Description
`combo_enable()`	Enables the combo feature
`combo_disable()`	Disables the combo feature, and clears the combo buffer
`combo_toggle()`	Toggles the state of the combo feature
`is_combo_enabled()`	Returns the status of the combo feature state (true or false)