`main.cpp` Contains the executable starting point, initialization code and the list of known PowerToys. All singletones are also initialized here at the start. Loads all the powertoys by scanning the `./modules` folder and `enable()`s those marked as enabled in `%LOCALAPPDATA%\Microsoft\PowerToys\settings.json` config. Then it runs a message loop for the tray UI. Note that this message loop also handles lowlevel_keyboard_hook events.

`powertoy_module.h` and `powertoy_module.cpp` Contains code for initializing and managing the PowerToy modules. `PowertoyModule` is a RAII-style holder for the `PowertoyModuleIface` pointer, which we got by invoking module DLL's `powertoy_create` function.

`powertoys_events.cpp` Contains code that handles the various events listeners, and forwards those events to the PowerToys modules. You can learn more about the current event architecture here.

`lowlevel_keyboard_event.cpp` Contains code for registering the low level keyboard event hook that listens for keyboard events. Please note that `signal_event` is called from the main thread for this event.

`win_hook_event.cpp` Contains code for registering a Windows event hook through `SetWinEventHook`, that listens for various events raised when a window is interacted with. Please note, that `signal_event` is called from a separate `dispatch_thread_proc` worker thread, so you must provide thread-safety for your `signal_event` if you intend to receive it. This is a subject to change.

`tray_icon.cpp` Contains code for managing the PowerToys tray icon and its menu commands. Note that `dispatch_run_on_main_ui_thread` is used to transfer received json message from the Settings window to the main thread, since we're communicating with it from a dedicated thread. #### `settings_window.cpp` Contains code for starting the PowerToys settings window and communicating with it. Settings window is a separate process, so we're using Windows pipes as a transport for json messages.

`general_settings.cpp` Contains code for loading, saving and applying the general settings.

`auto_start_helper.cpp` Contains helper code for registering and unregistering PowerToys to run when the user logs in.

`unhandled_exception_handler.cpp` Contains helper code to get stack traces in builds. Can be used by adding a call to `init_global_error_handlers` in `WinMain`.

`trace.cpp` Contains code for telemetry.

`svgs` Contains the SVG assets used by the PowerToys modules.